



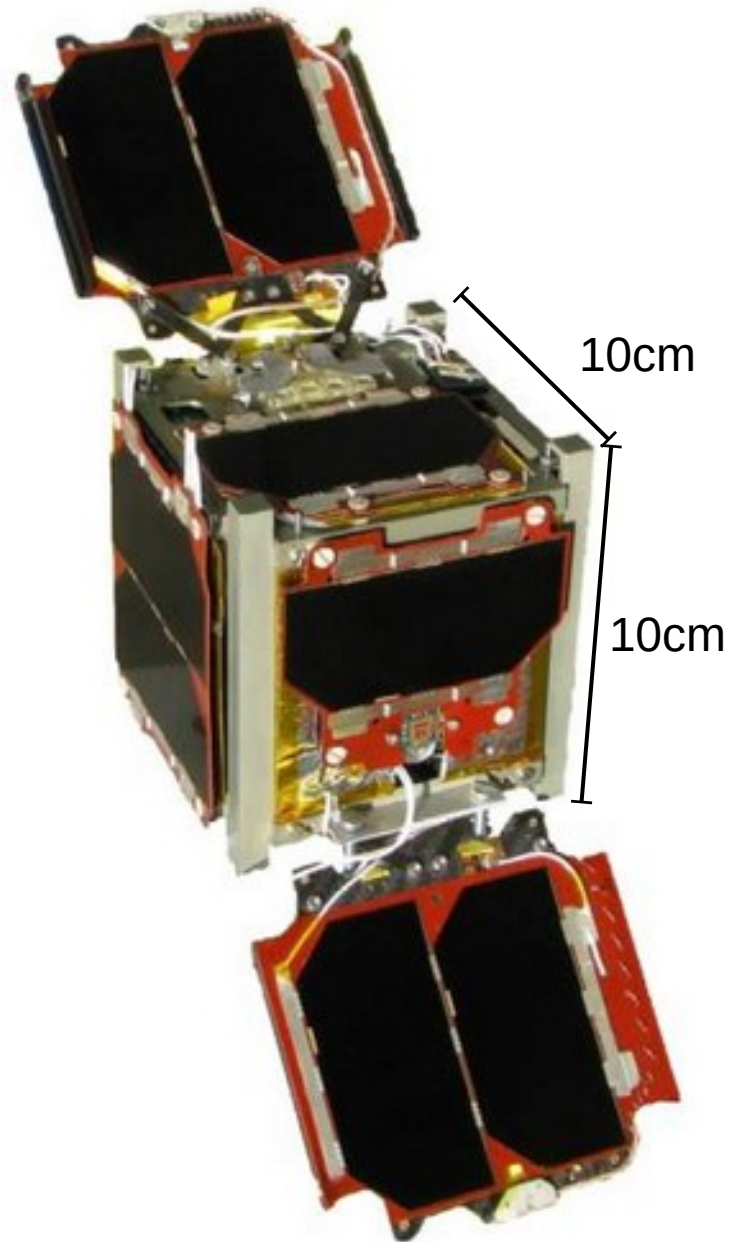
Fault-Tolerant Satellite Computing on a Budget

Christian M. Fuchs

Leiden University



NPI 497-2016



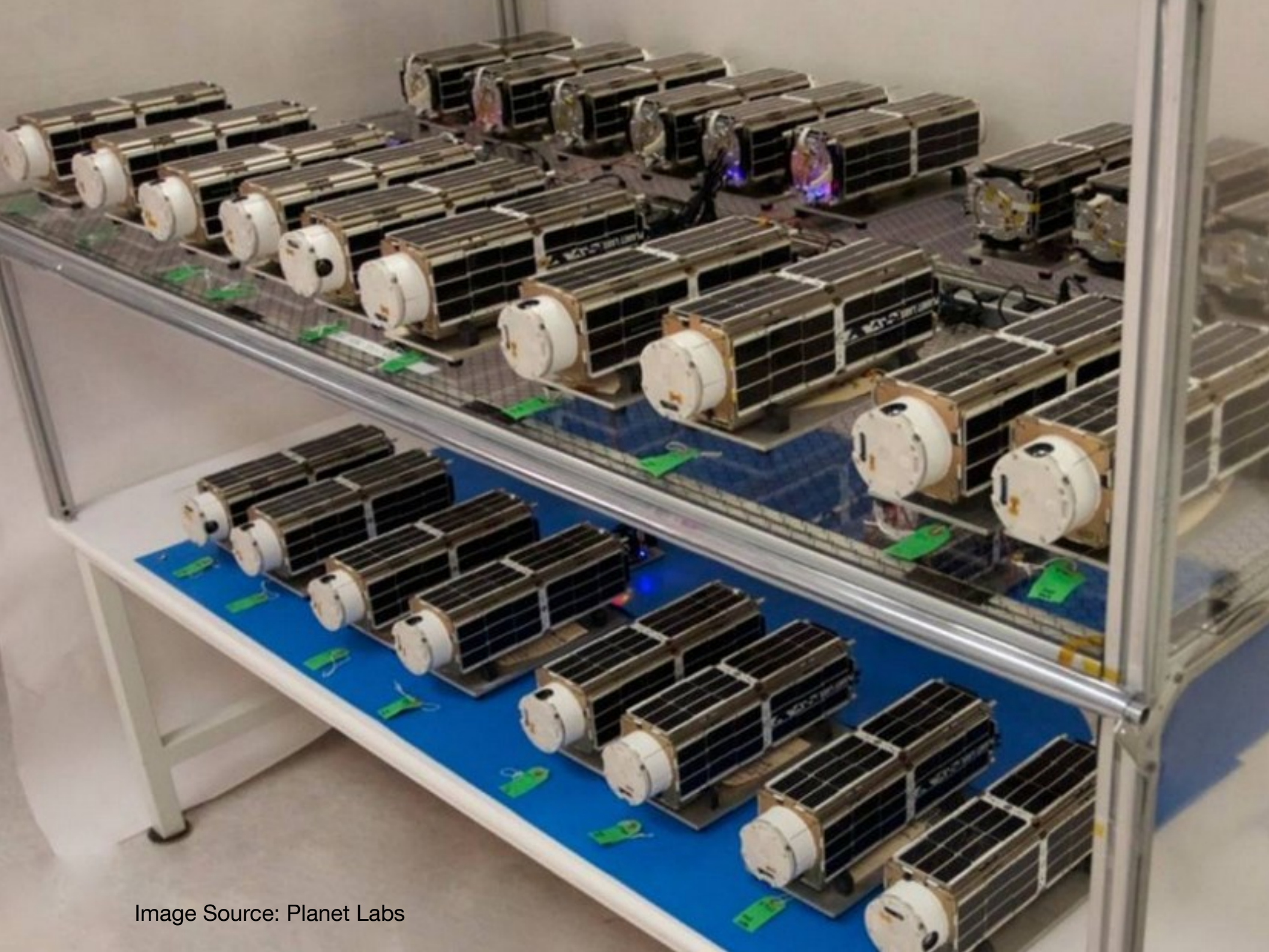


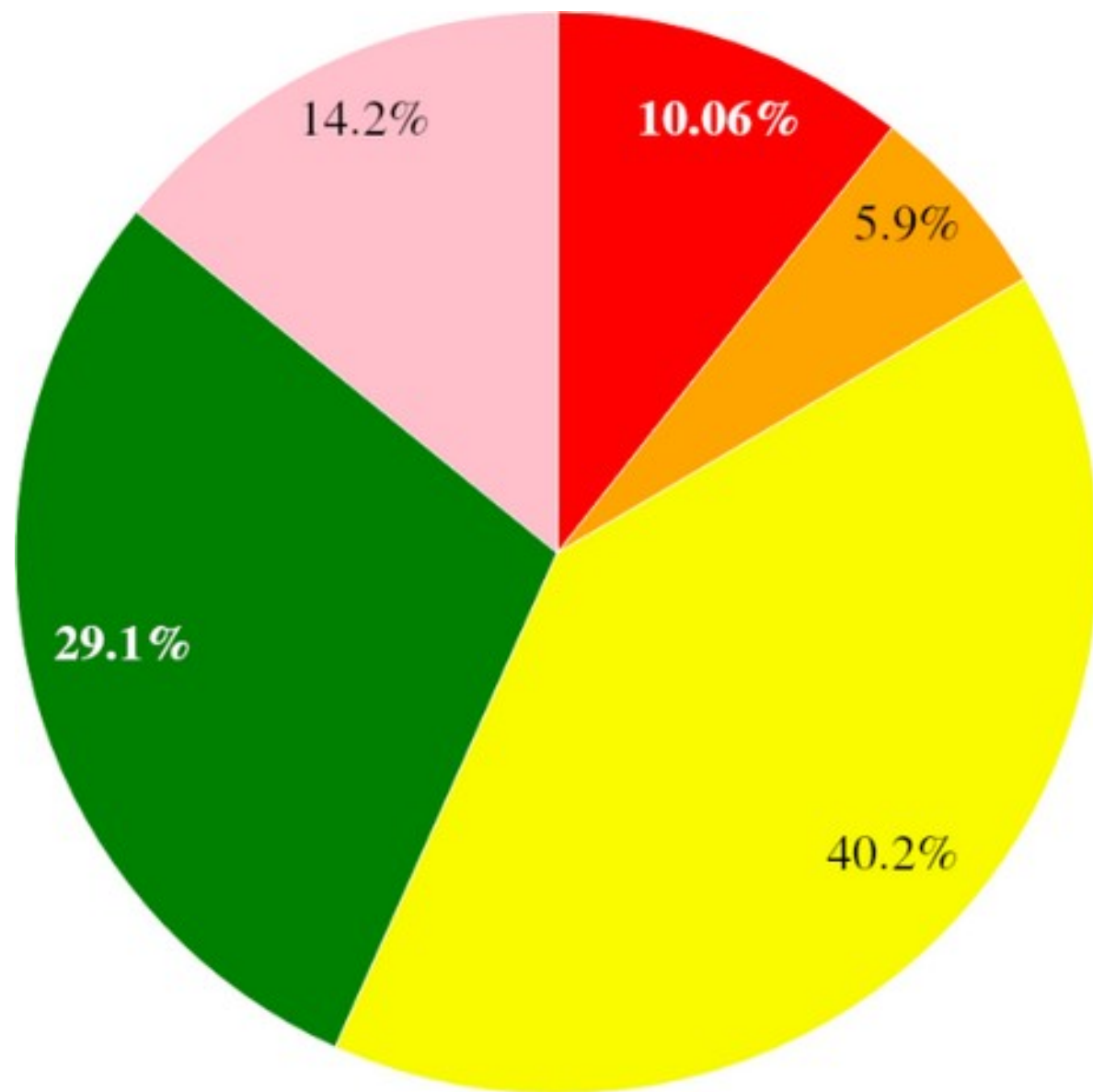
Image Source: Planet Labs

Nanosatellite Hardware





CubeSat Missions Success



- Full Mission Success
- Partial Mission Success
- Early Failure
- Dead on Arrival
- No Data/Unknown

Documented Launches

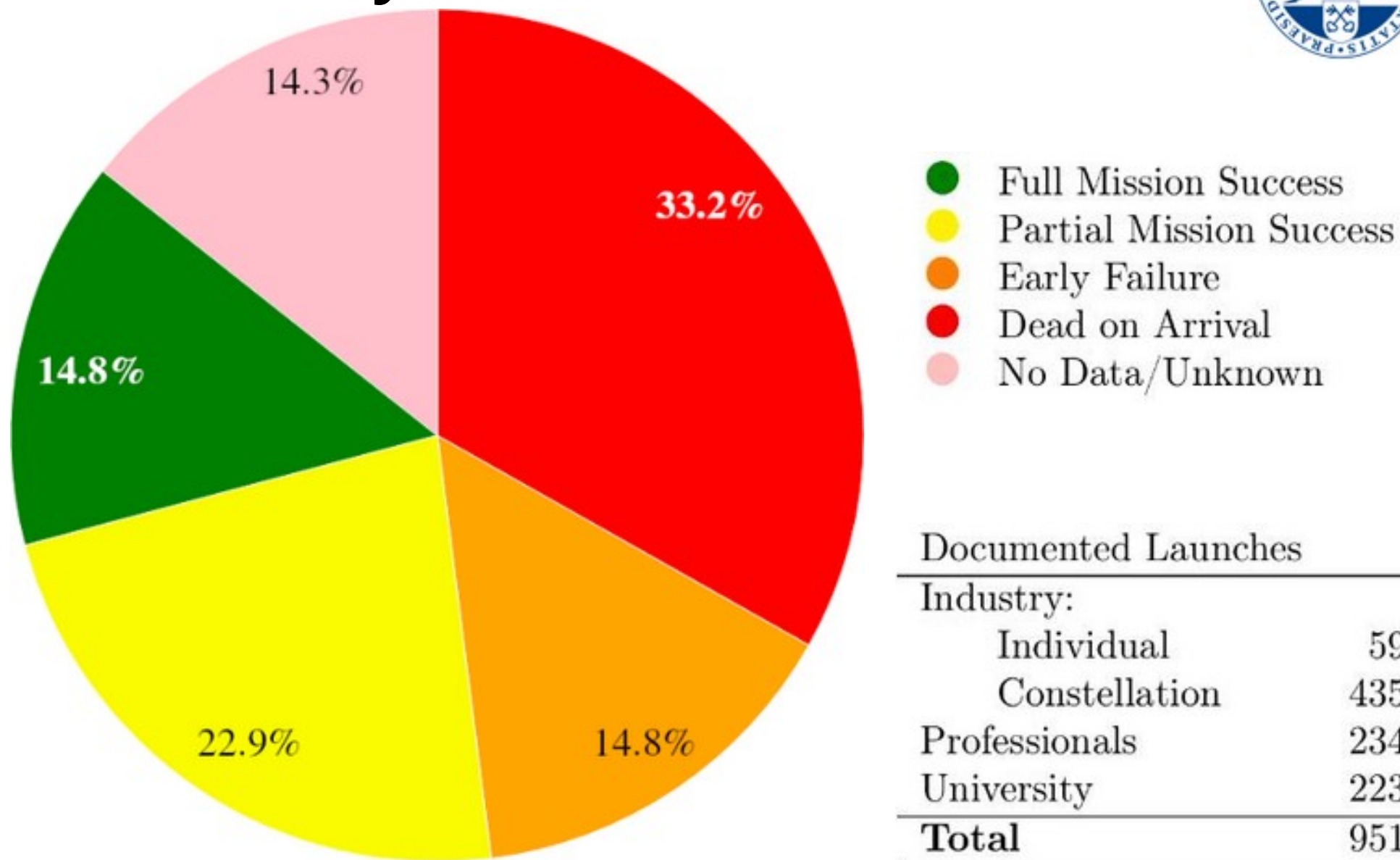
Industry:

Individual	59
Constellation	435
Professionals	234
University	223
Total	951

(Data by M. Swartwout, 2018)



University Missions Success

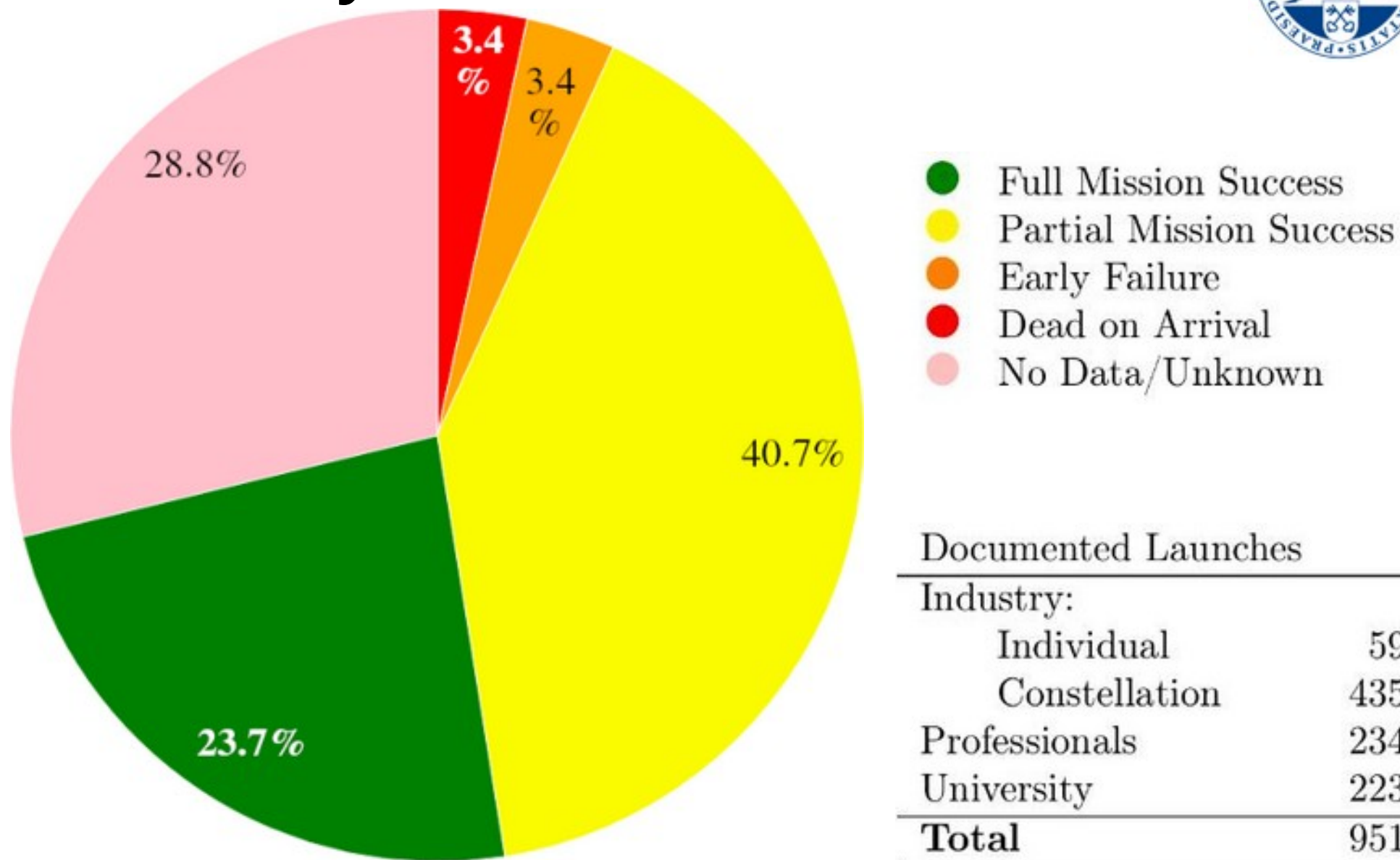


(Data by M. Swartwout, 2018)

Documented Launches	
Industry:	
Individual	59
Constellation	435
Professionals	234
University	223
Total	951



Industry Missions Success



(Data by M. Swartwout, 2018)

Documented Launches	
Industry:	
Individual	59
Constellation	435
Professionals	234
University	223
Total	951

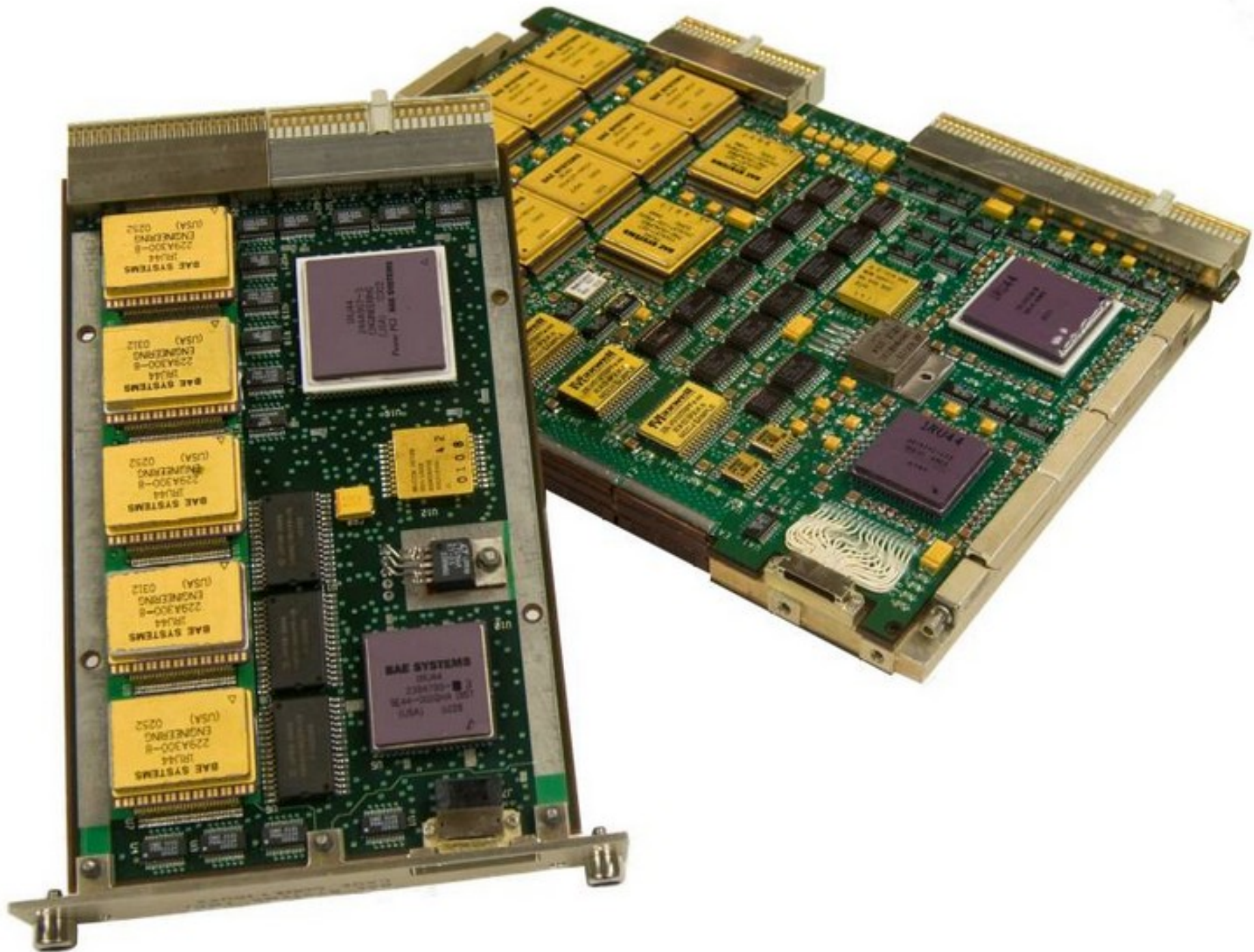


Image Source: BAE Systems

Design Lifetime

Space Grade Hardware

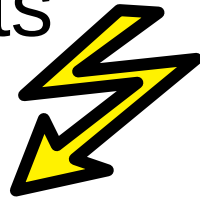
- 10y+ by Design
- 30y+ Best Case



Design Lifetime

Today's CubeSats

- 0y by Design
- 10y Best Case



Design Lifetime

Today's CubeSats

- **0y by Design**
- 10y Best Case



Space Grade Hardware

- 10y+ by Design
- 30y+ Best Case



Design Lifetime

Future CubeSats

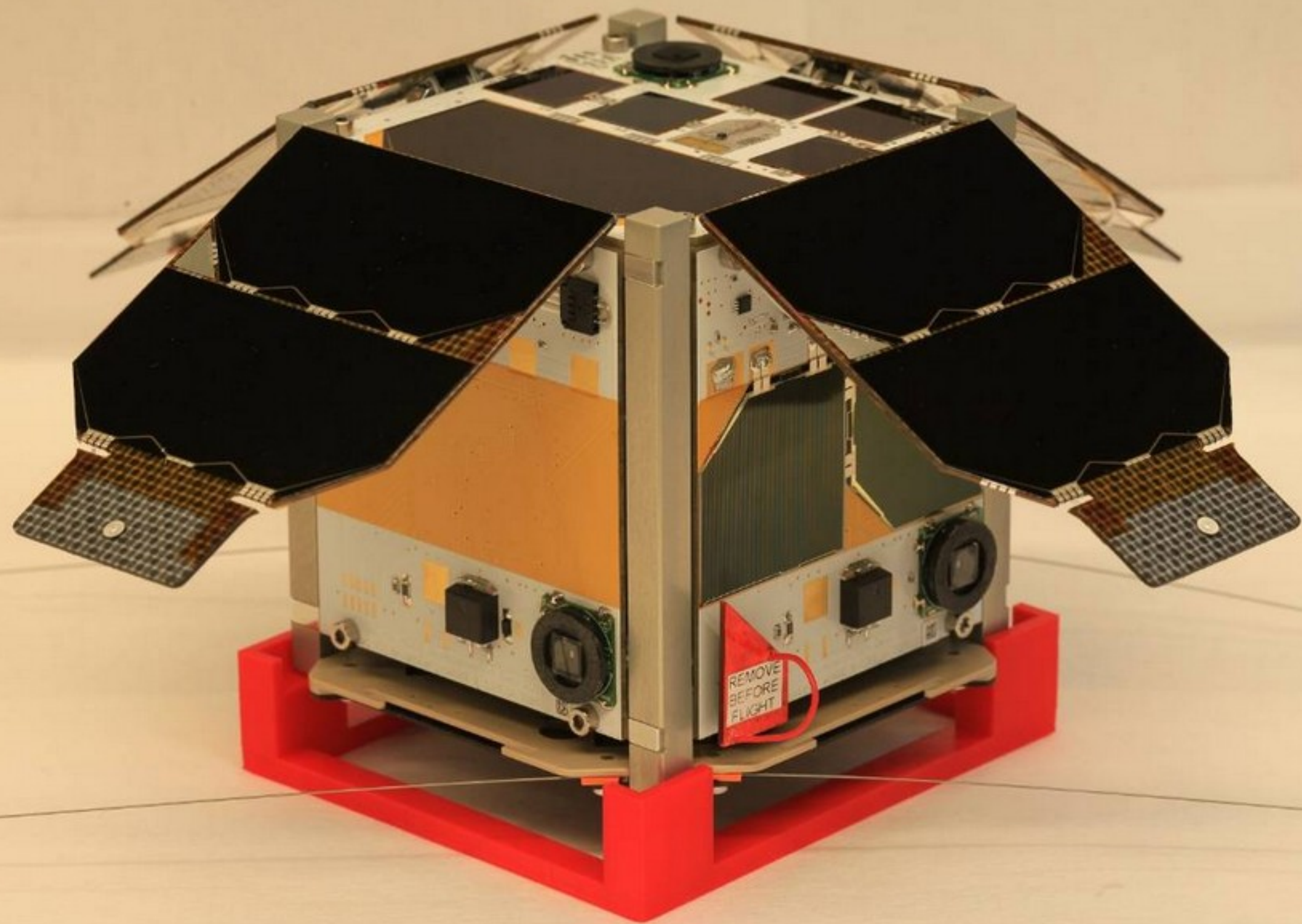
- **2-5y by Design**
- 10y Best Case



Space Grade Hardware

- 10y by Design
- 30y Best Case







How to Achieve Fault Tolerance

- **“The Classics”**
 - Proven legacy technology
 - Custom space-grade components
- “The Holy Grail”
 - Radiation-immune Technologies
 - Robust microfabrication as standard
- Redundancy and Majority Voting
 - TMR for Logic, Cores, or Components in Hardware
 - Custom instruction sets or instruction pipelines
- Erasure Coding
- Software Measures – in Theory



How to Achieve Fault Tolerance

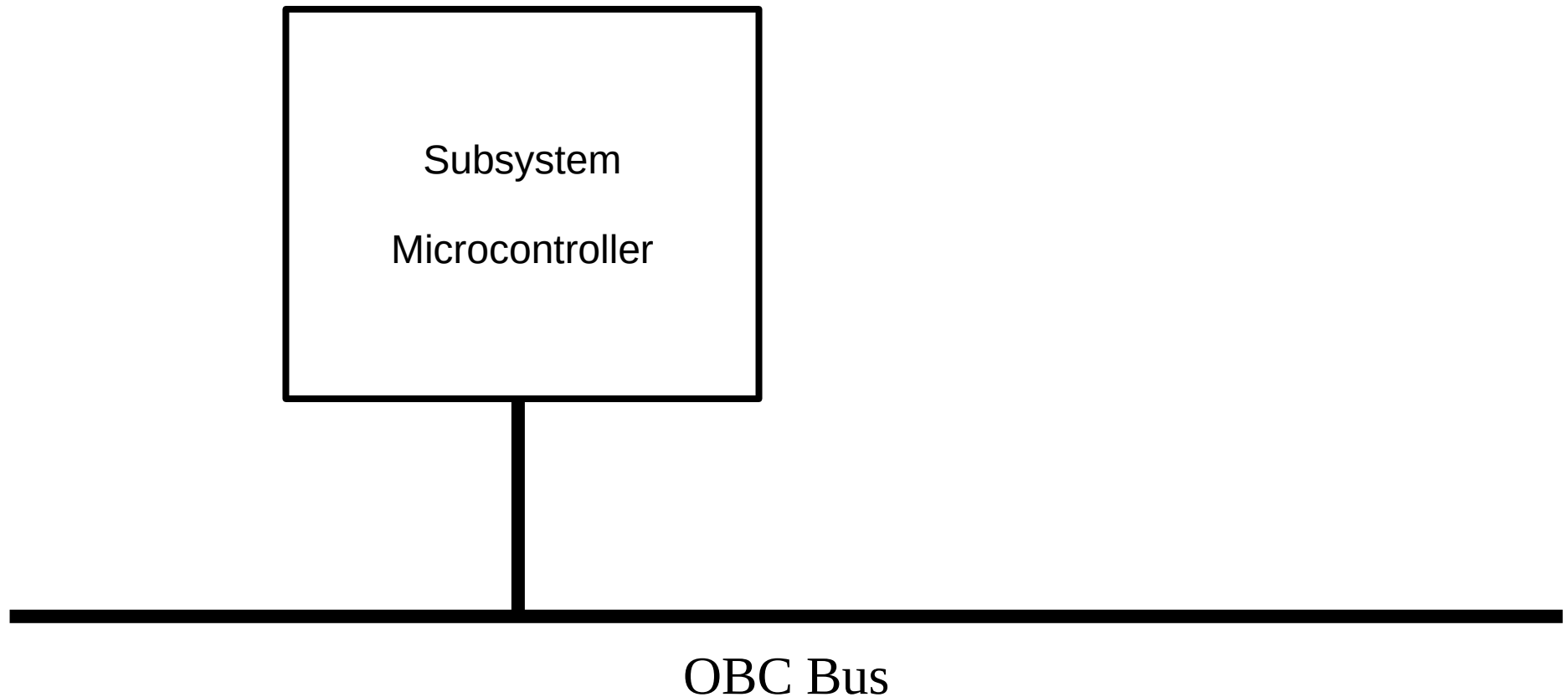
- ~~“The Classics”~~
 - ~~Proven legacy technology~~
 - ~~Custom space-grade components~~
- **“The Holy Grail”**
 - Radiation-immune Technologies
 - Robust microfabrication as standard
- Redundancy and Majority Voting
 - TMR for Logic, Cores, or Components in Hardware
 - Custom instruction sets or instruction pipelines
- Erasure Coding
- Software Measures – in Theory



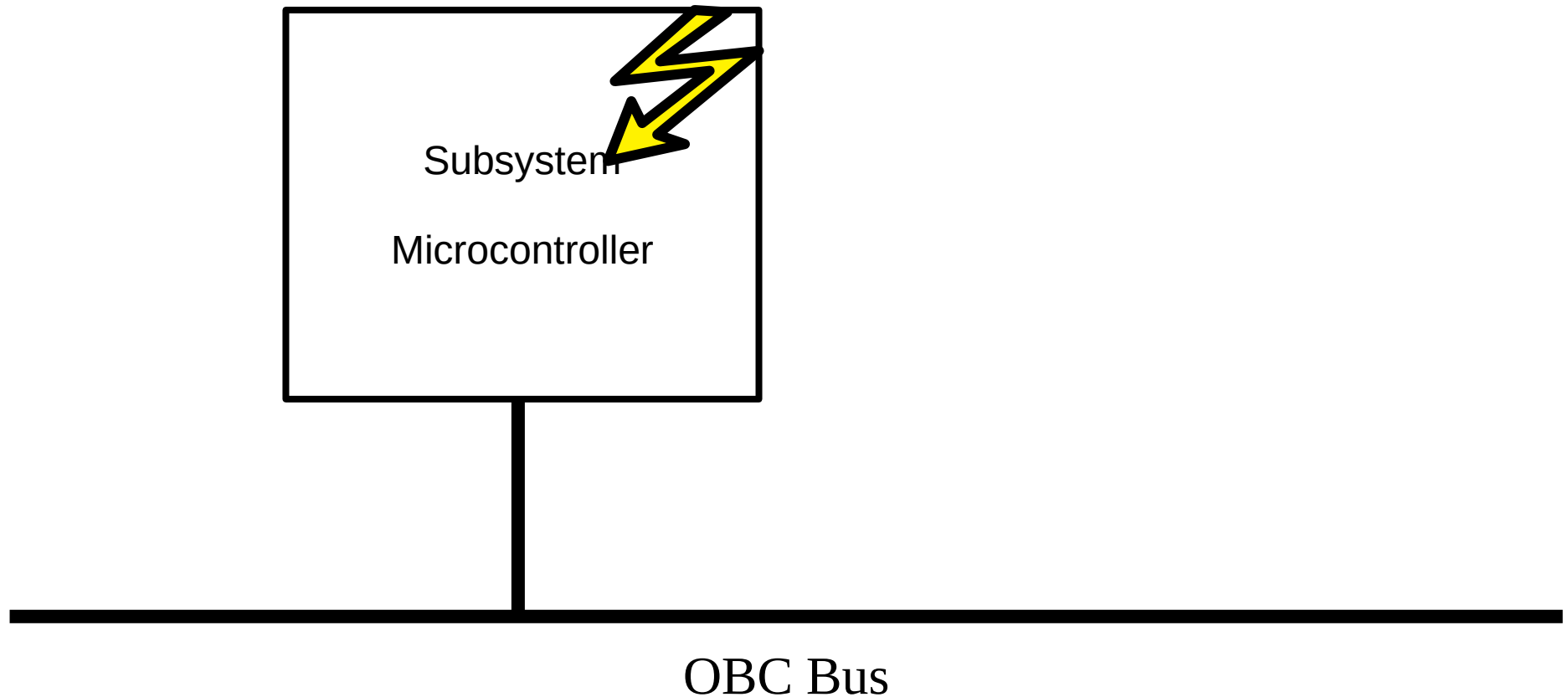
How to Achieve Fault Tolerance

- ~~“The Classics”~~
 - ~~Proven legacy technology~~
 - ~~Custom space-grade components~~
- ~~“The Holy Grail”~~
 - ~~Radiation-immune Technologies~~
 - ~~Robust microfabrication as standard~~
- **Redundancy and Majority Voting**
 - TMR for Logic, Cores, or Components in Hardware
 - Custom instruction sets or instruction pipelines
- Erasure Coding
- Software Measures – in Theory

Component Level Redundancy

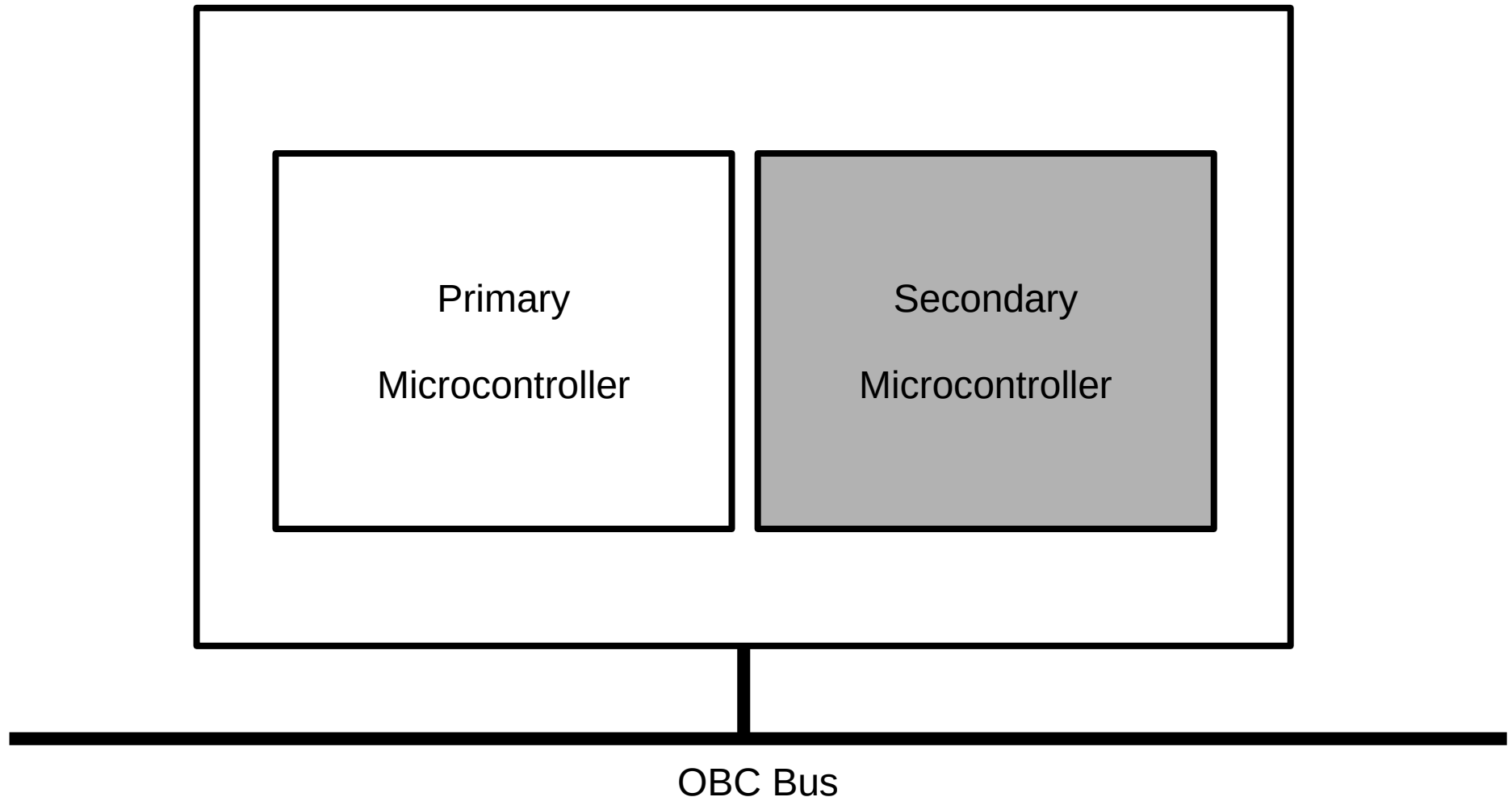


Component Level Redundancy



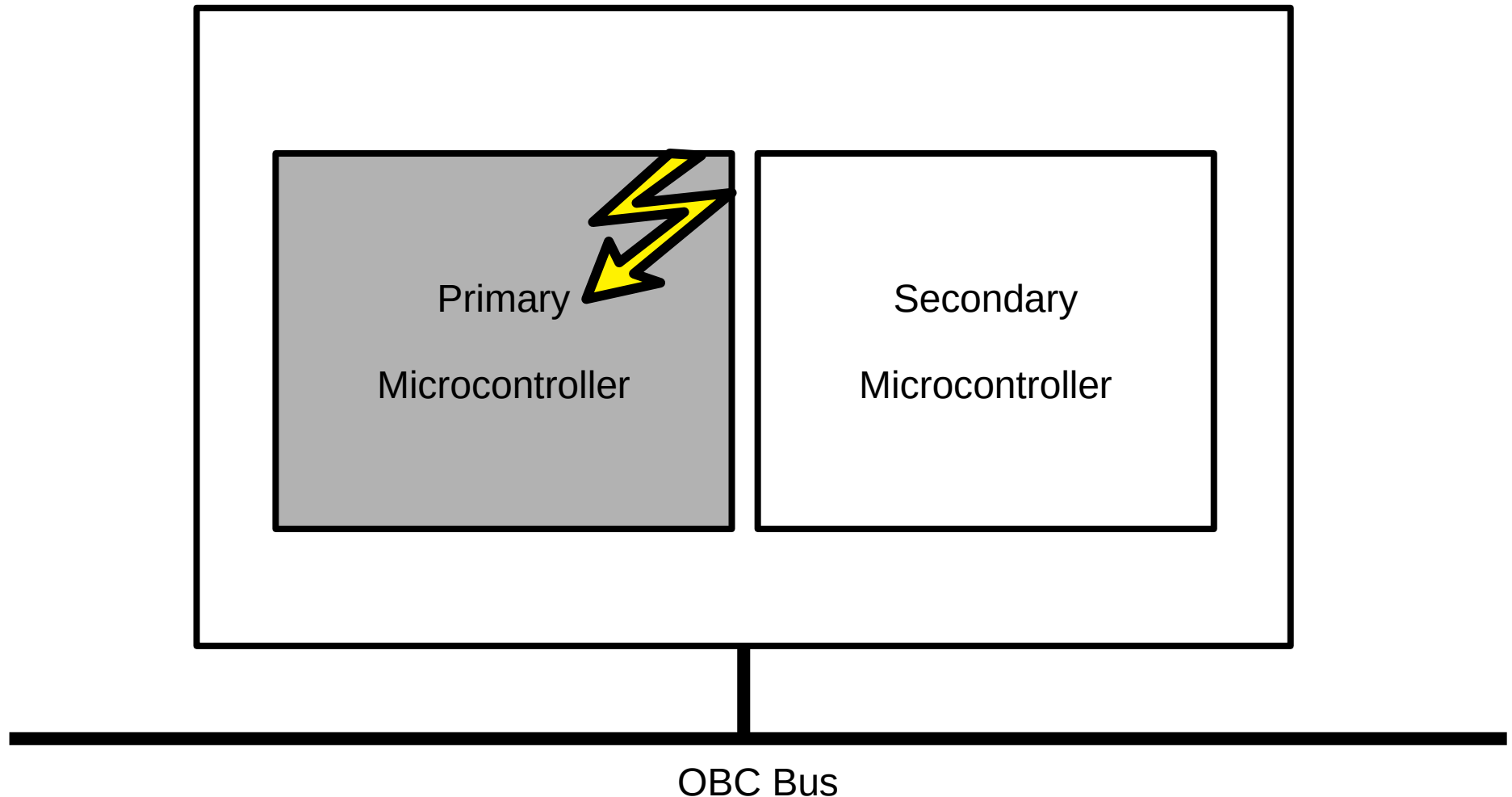
Component Level Redundancy

Subsystem Node



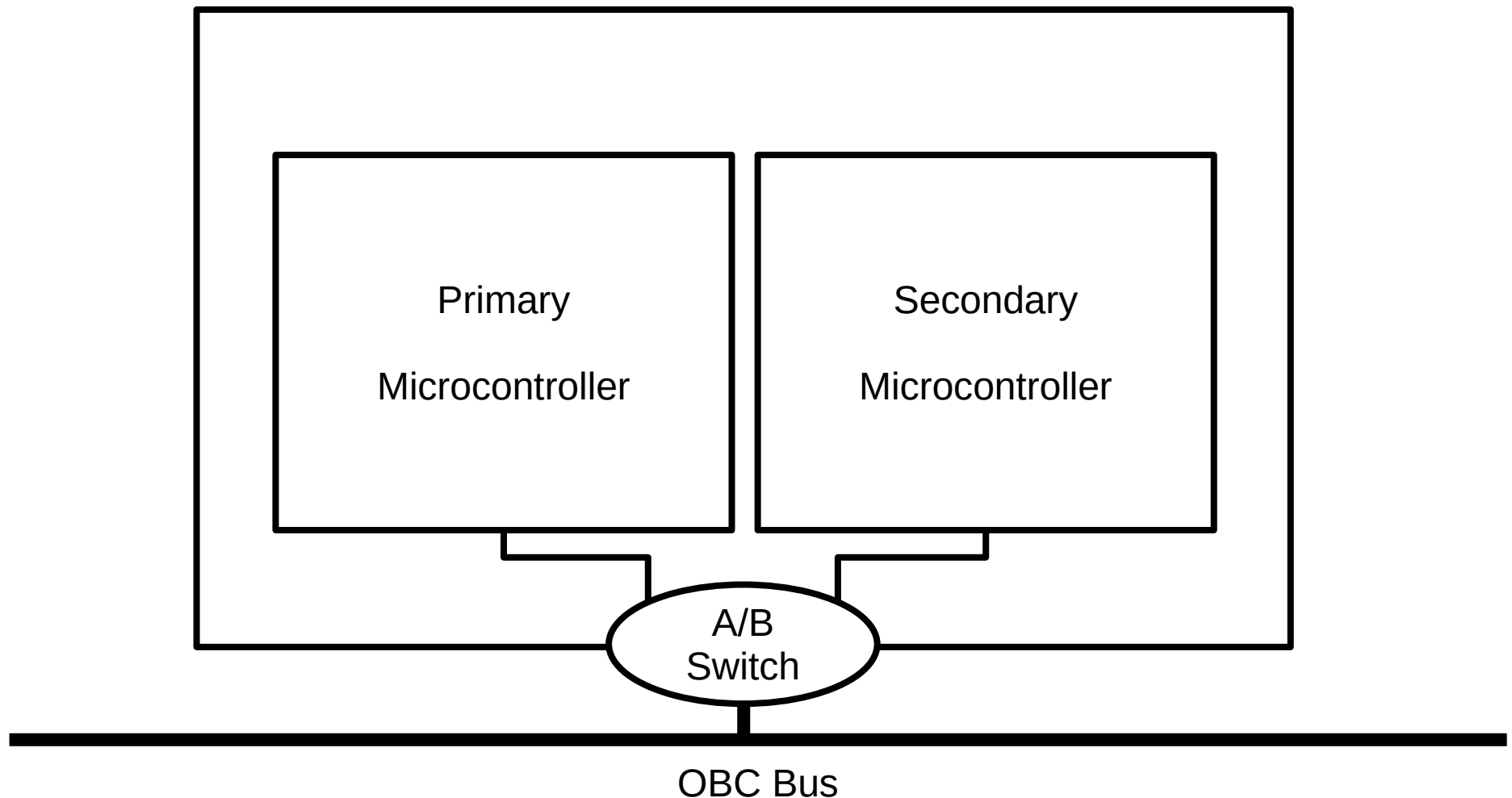
Component Level Redundancy

Subsystem Node



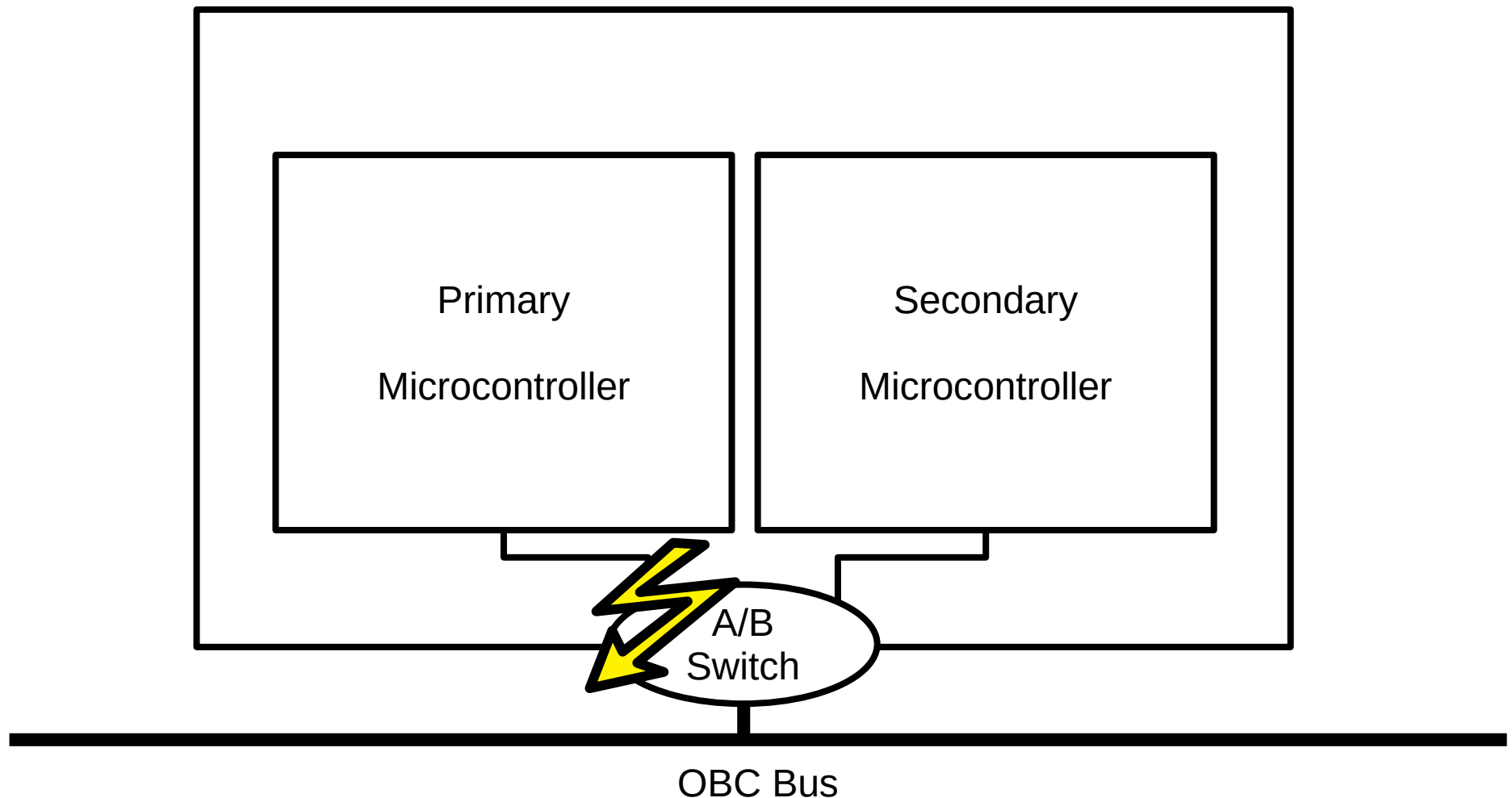
The Redundancy Fallacy

Subsystem Node

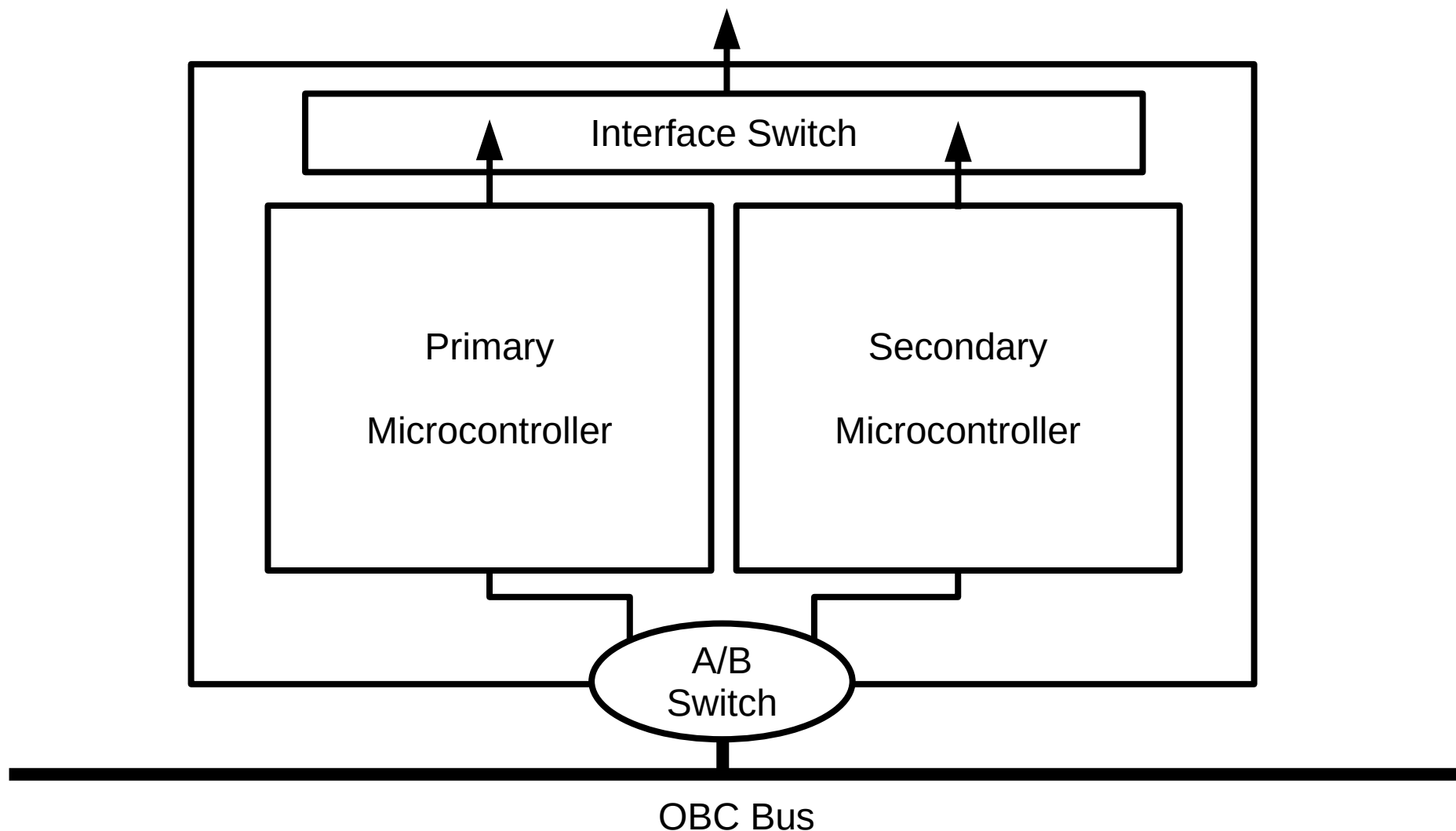


The Redundancy Fallacy

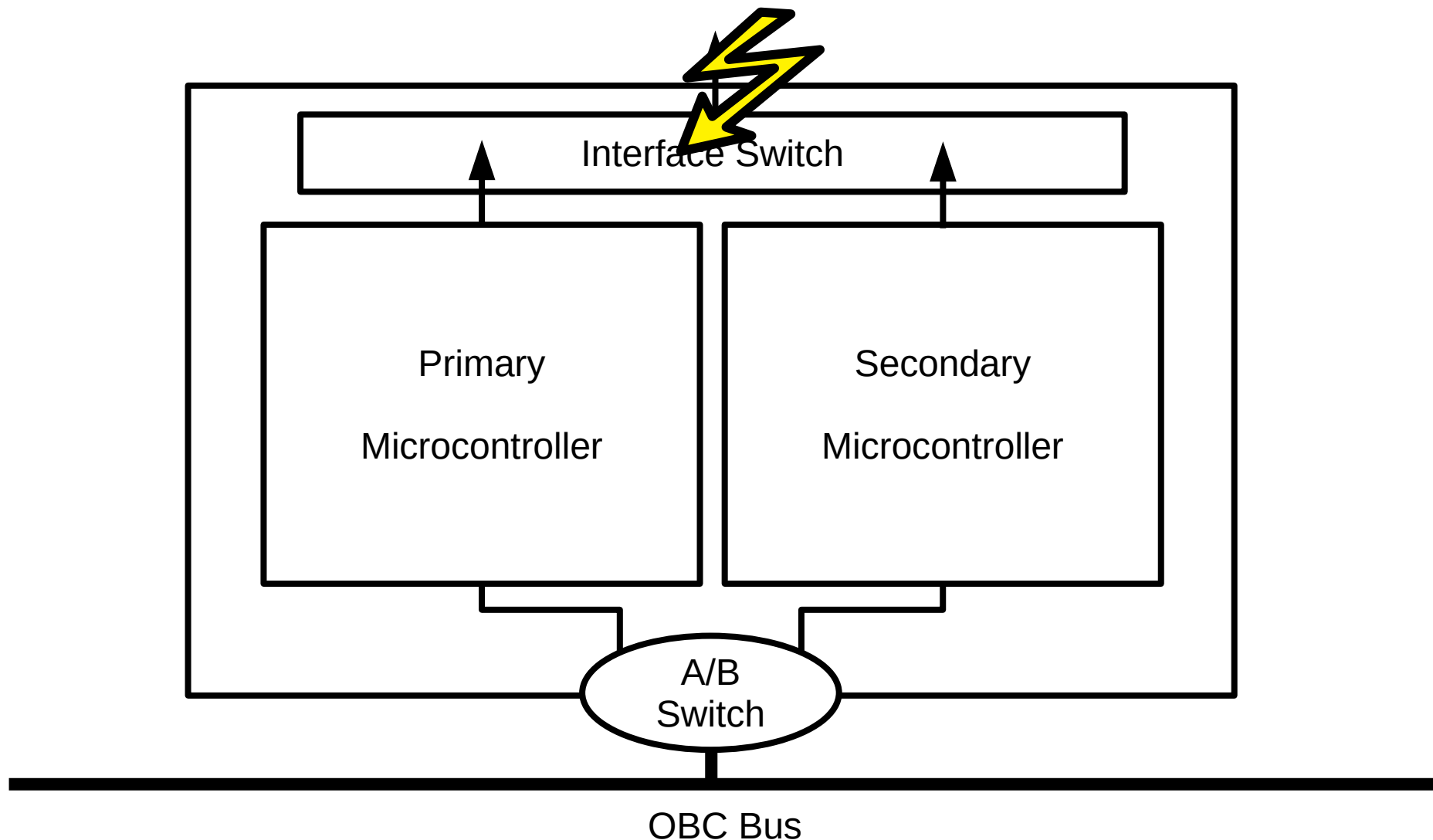
Subsystem Node



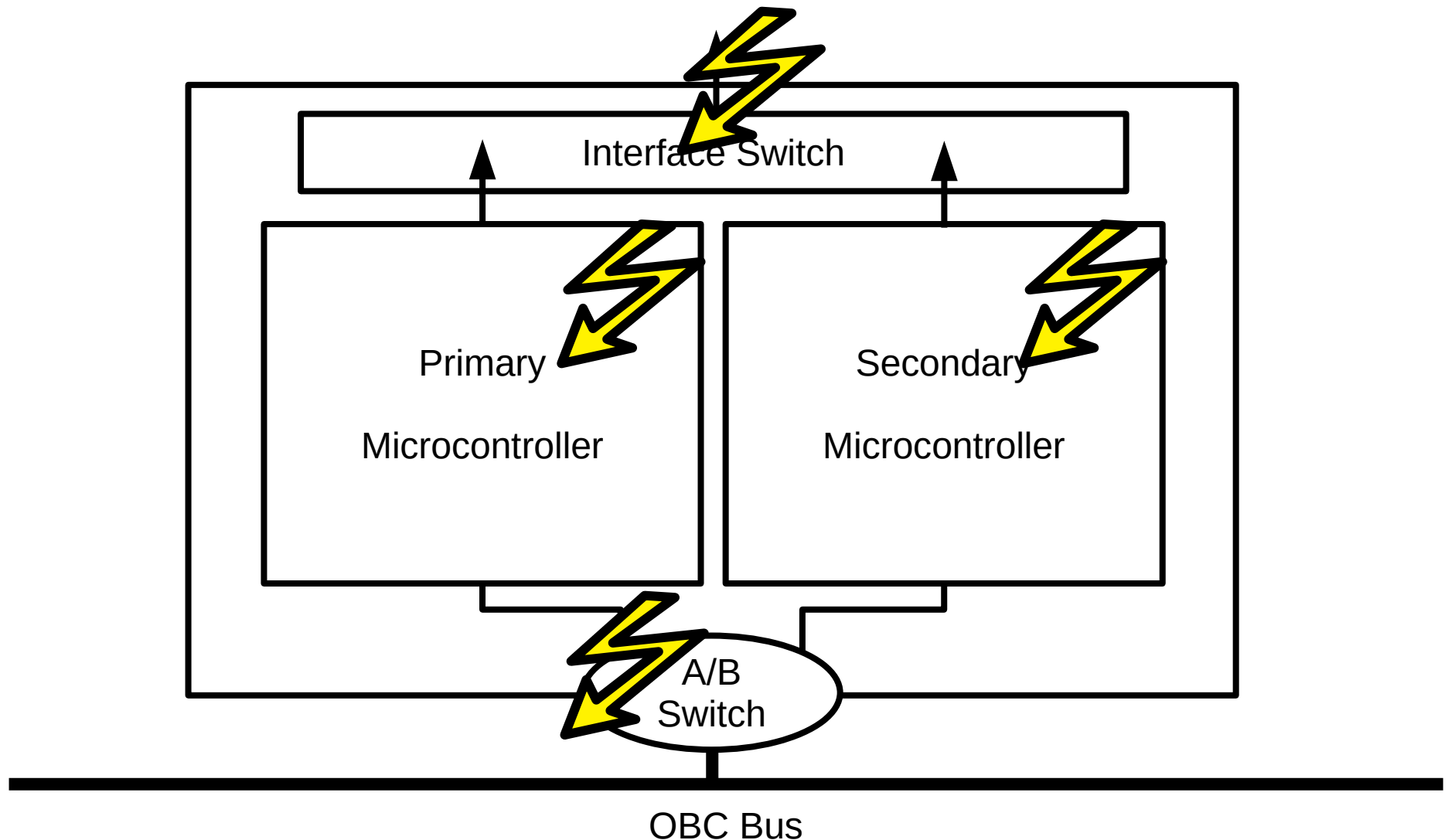
The Redundancy Fallacy



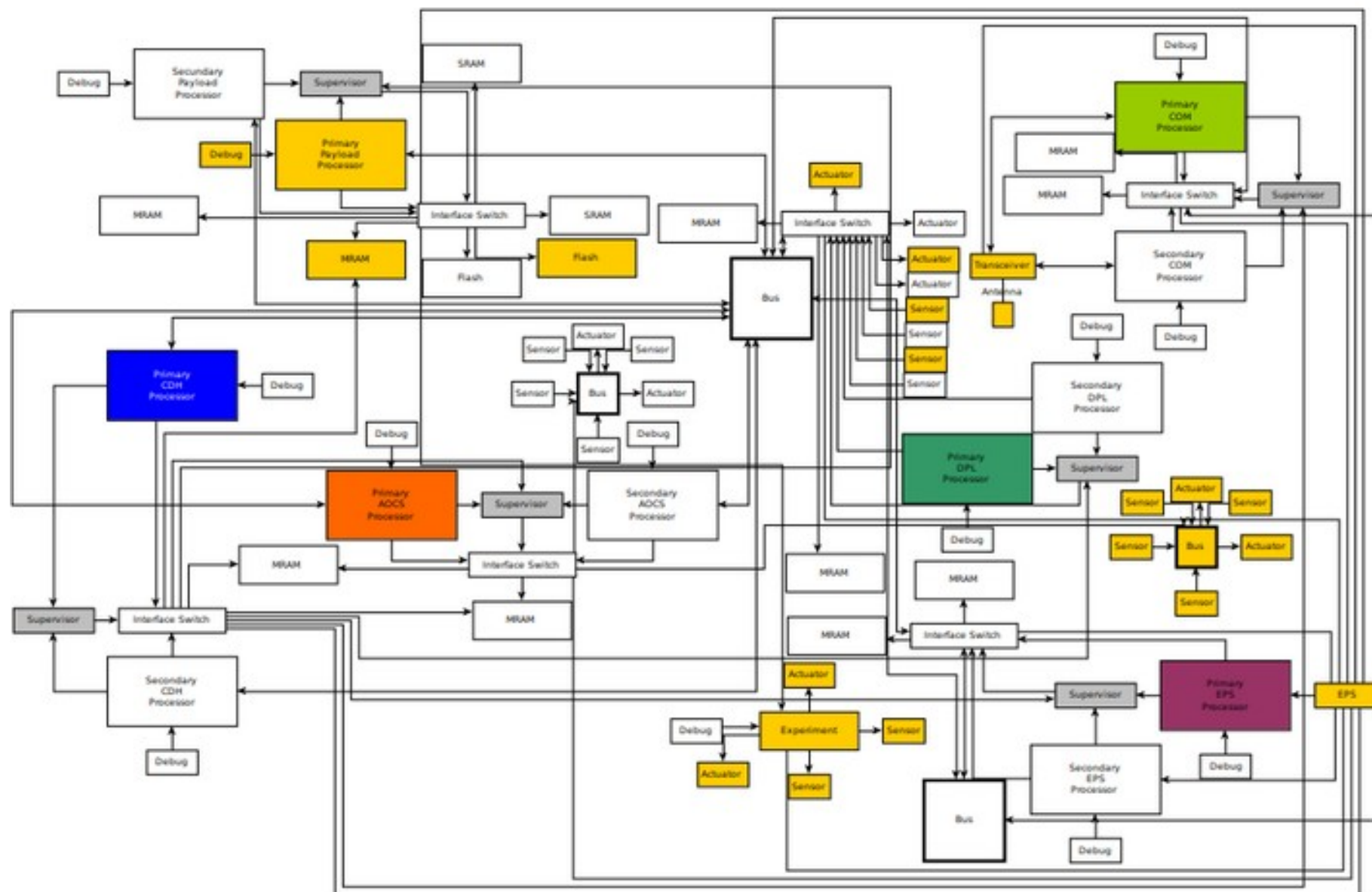
The Redundancy Fallacy



The Redundancy Fallacy



Limitations



How to Achieve Fault Tolerance

- ~~“The Classics”~~
 - ~~Proven legacy technology~~
 - ~~Custom space-grade components~~
- ~~“The Holy Grail”~~
 - ~~Radiation-immune Technologies~~
 - ~~Robust microfabrication as standard~~
- ~~Redundancy and Majority Voting~~
 - ~~TMR for Logic, Cores, or Components in Hardware~~
 - ~~Custom instruction sets or instruction pipelines~~
- **Erasur Coding**
- Software Measures – in Theory

How to Achieve Fault Tolerance

- ~~“The Classics”~~
 - ~~Proven legacy technology~~
 - ~~Custom space-grade components~~
- ~~“The Holy Grail”~~
 - ~~Radiation-immune Technologies~~
 - ~~Robust microfabrication as standard~~
- ~~Redundancy and Majority Voting~~
 - ~~TMR for Logic, Cores, or Components in Hardware~~
 - ~~Custom instruction sets or instruction pipelines~~
- ~~Erasure Coding~~
- **Software Measures – in Theory**

Design Lifetime

Future CubeSats

- **2-5y by Design**
- 10y Best Case

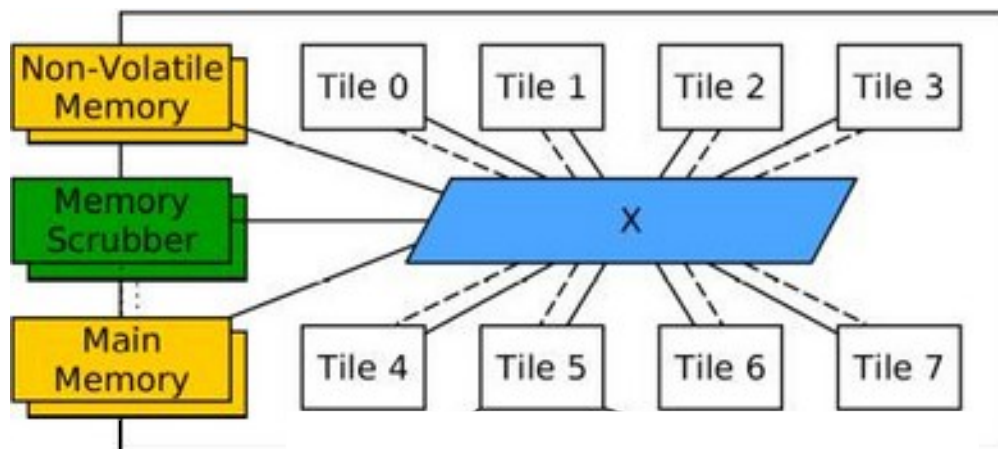


Space Grade Hardware

- 10y by Design
- 30y Best Case



Fault-Tolerance Architecture



1. Stage: Coarse Grained Lockstep
Short-term
2. Stage: Core-Recovery & Partial Reconfiguration
Medium-term
3. Stage: Graceful Aging through Mixed-Criticality
Long-term

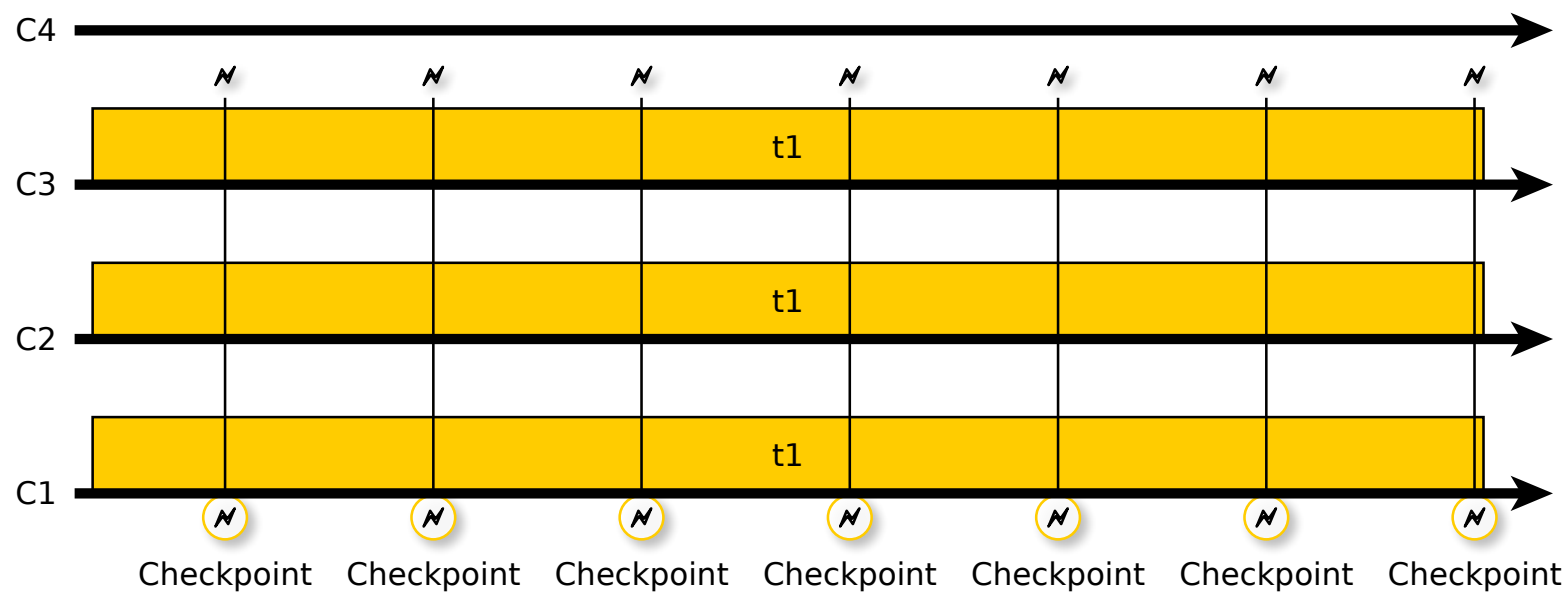
Software



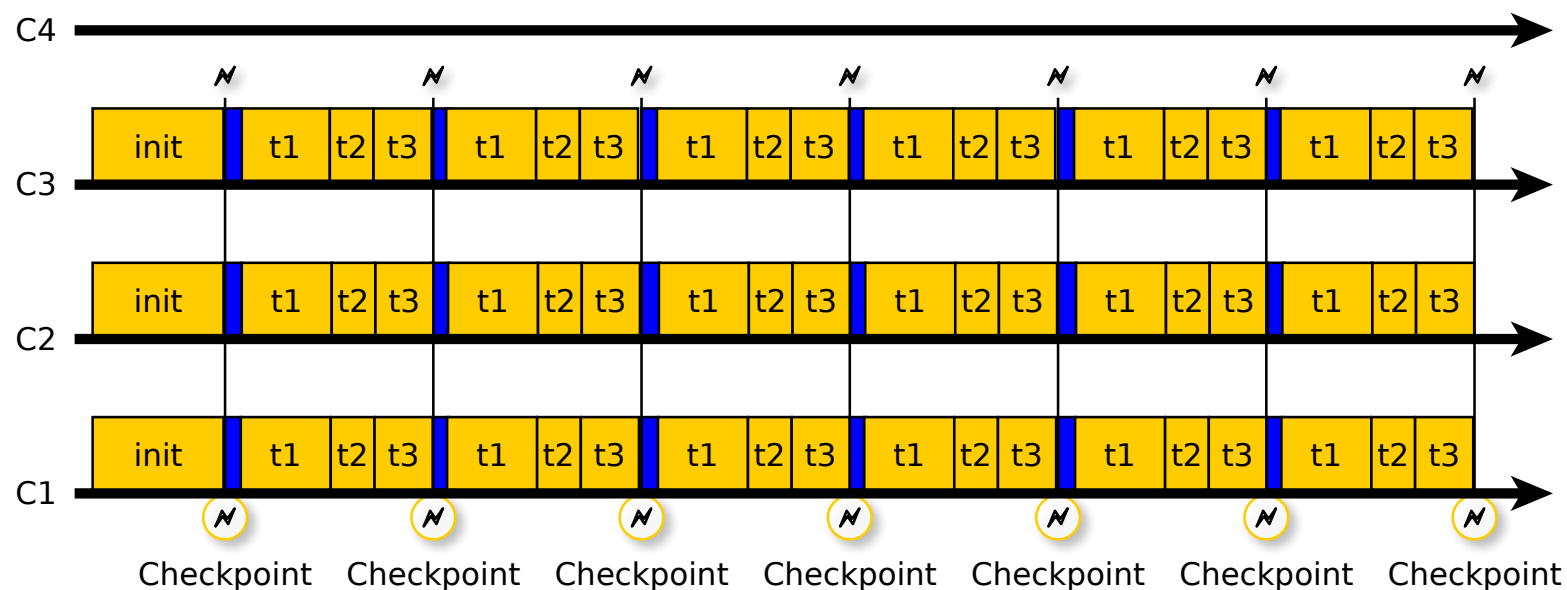
Software Redundancy



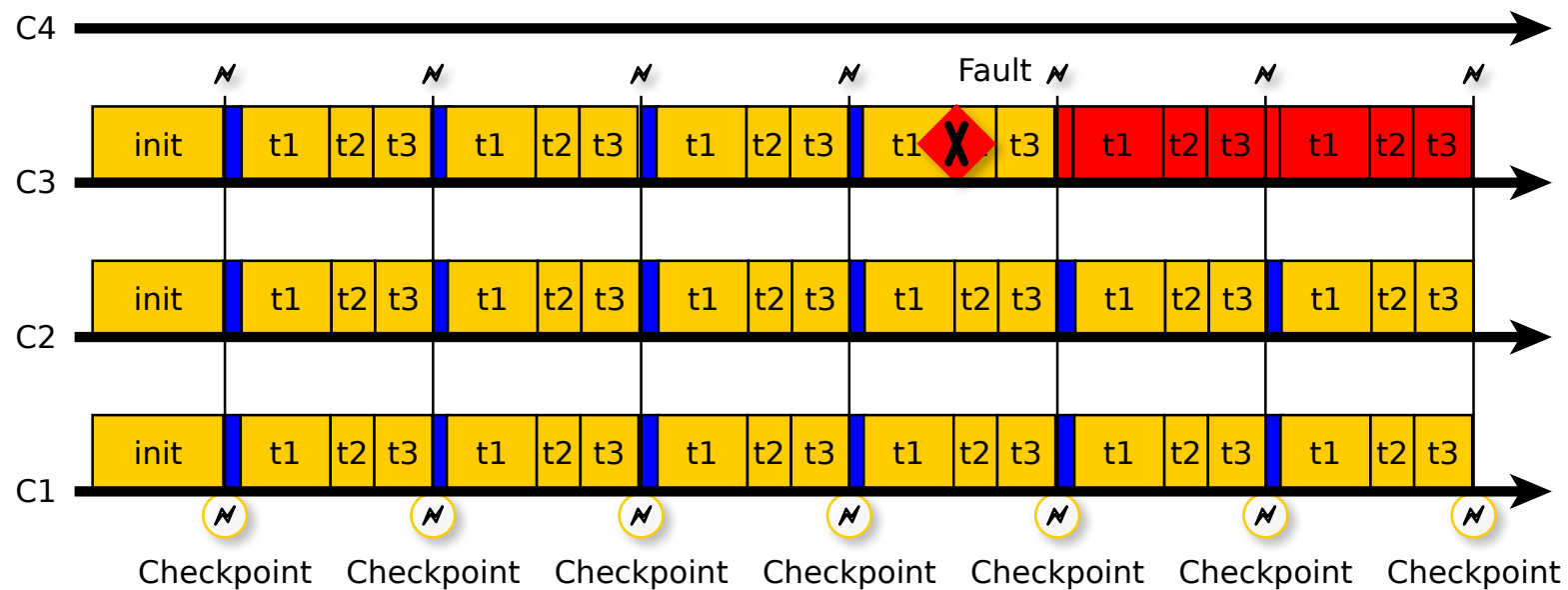
Coarse Grain Lockstep



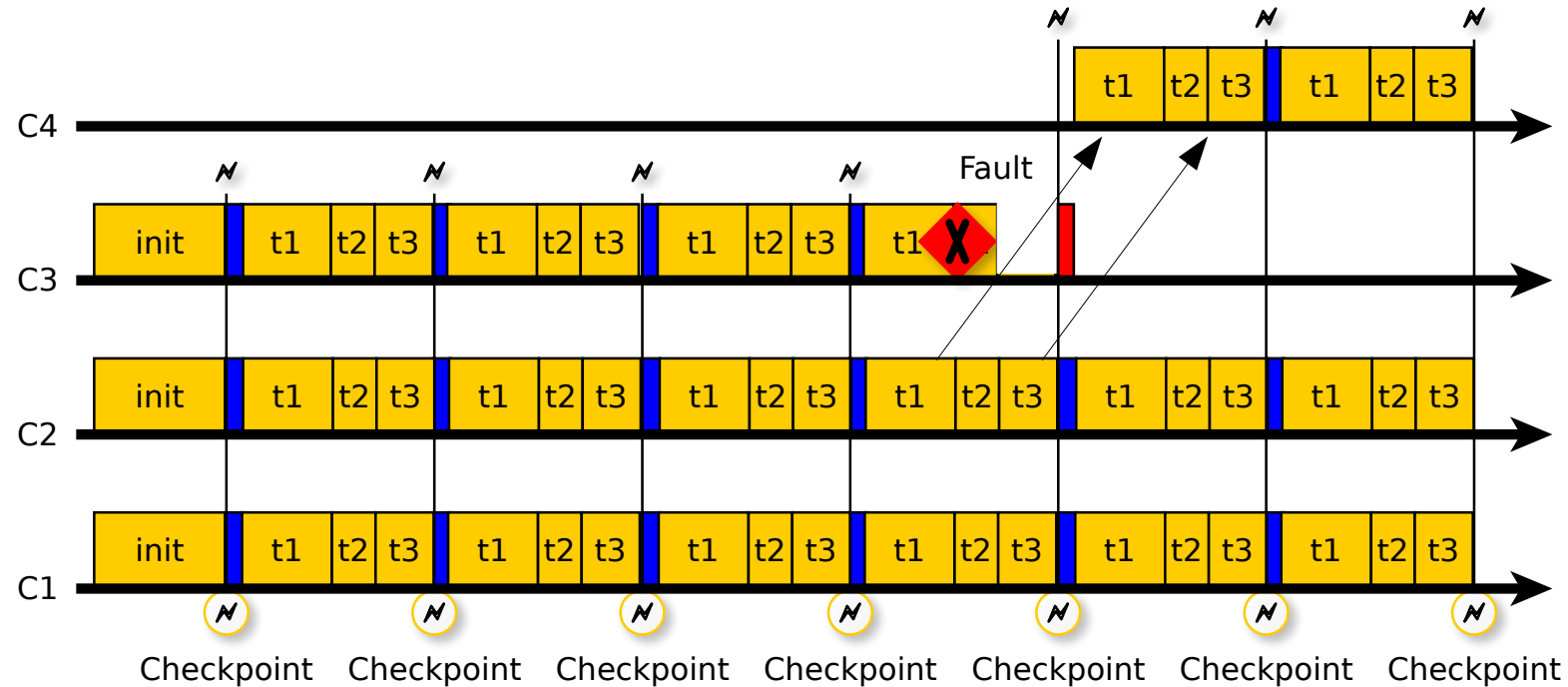
Coarse Grain Lockstep



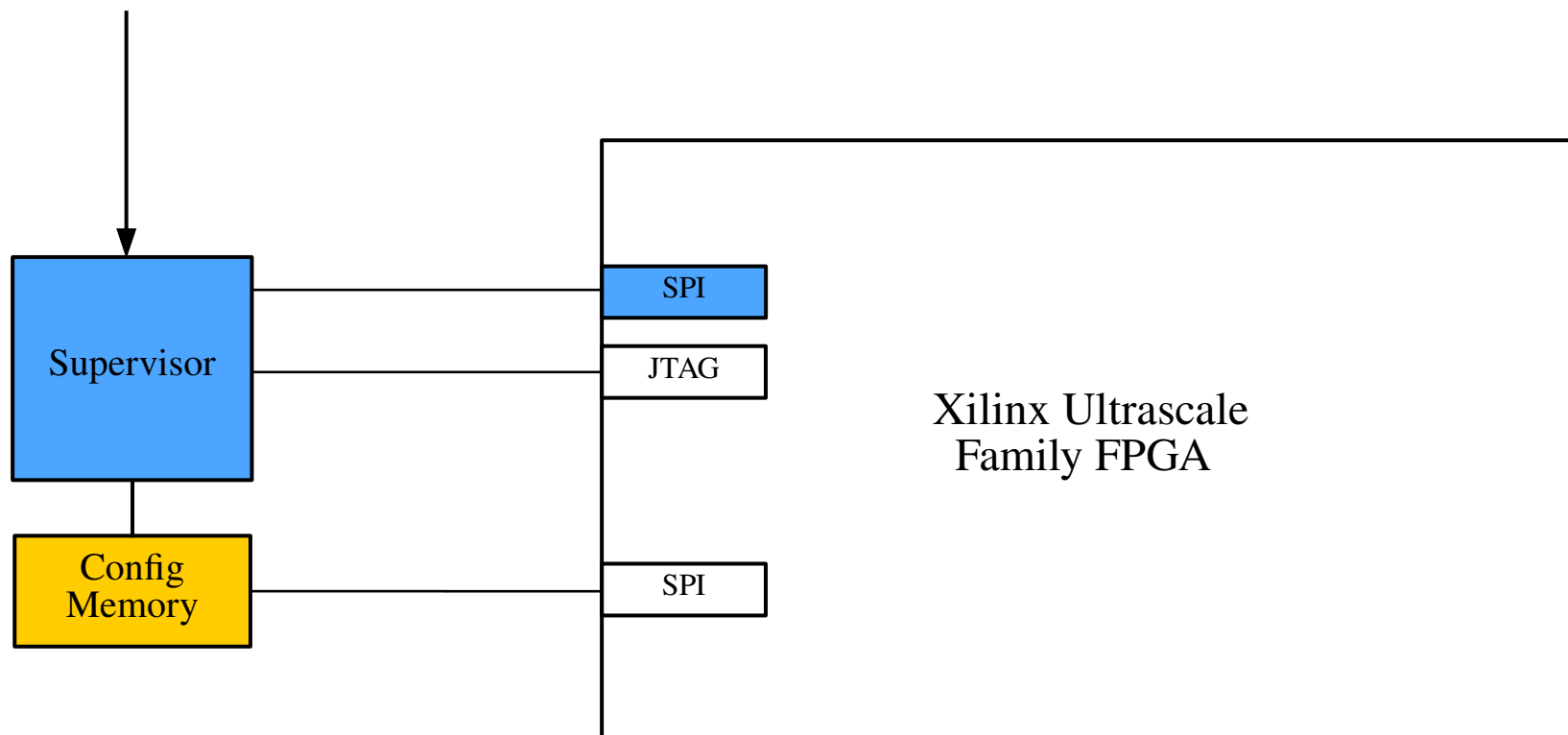
Coarse Grain Lockstep



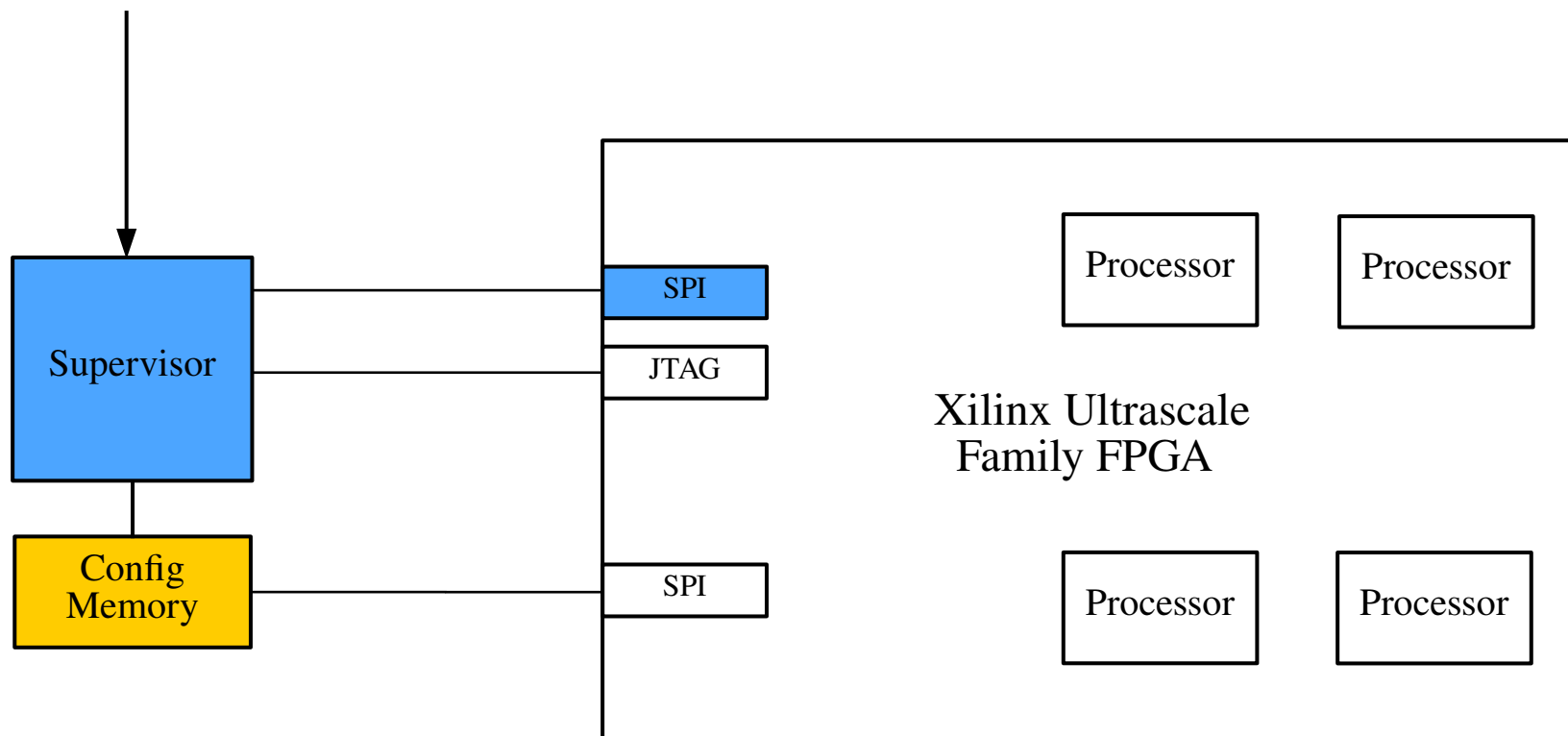
Coarse Grain Lockstep



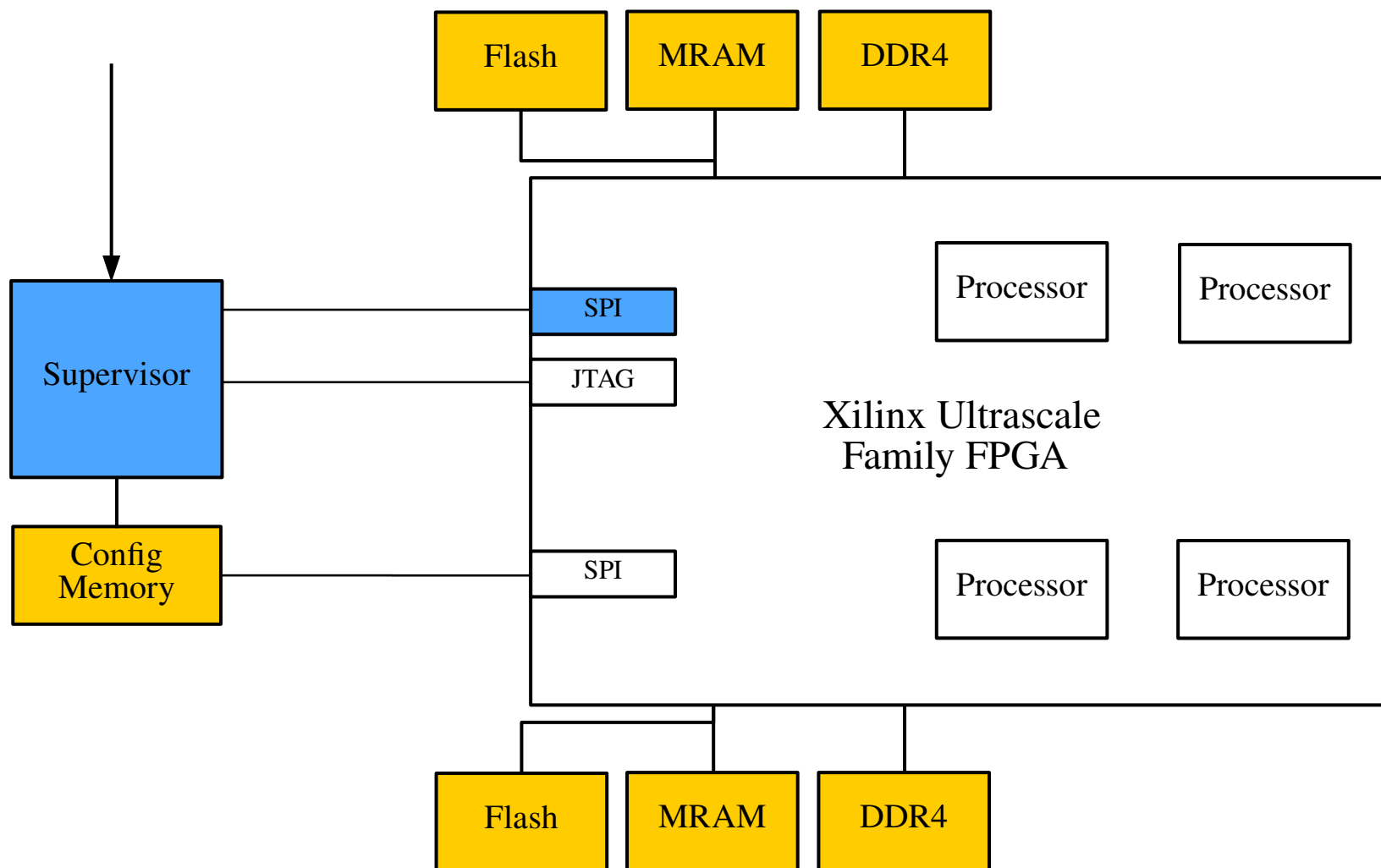
System Level View



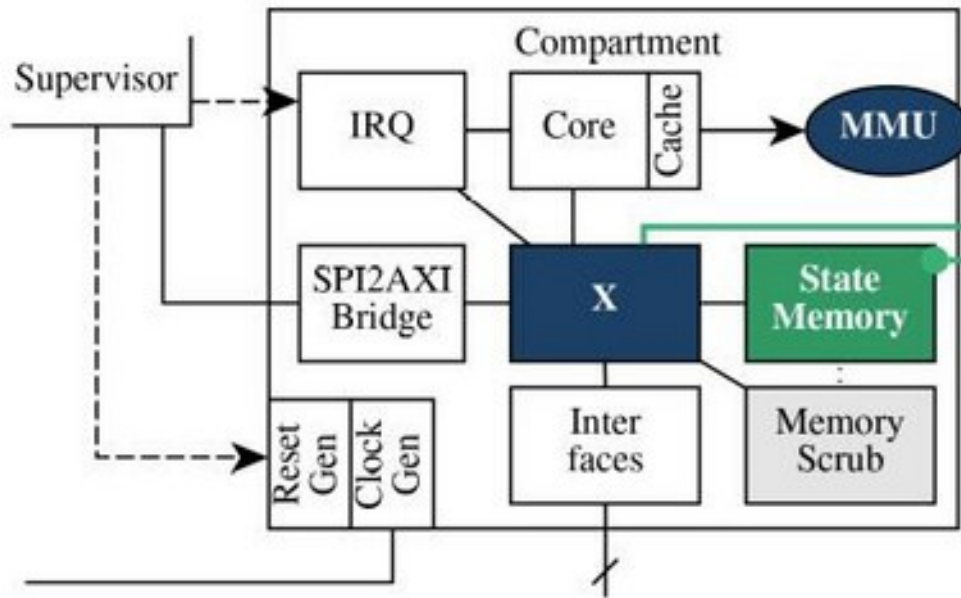
System Level View



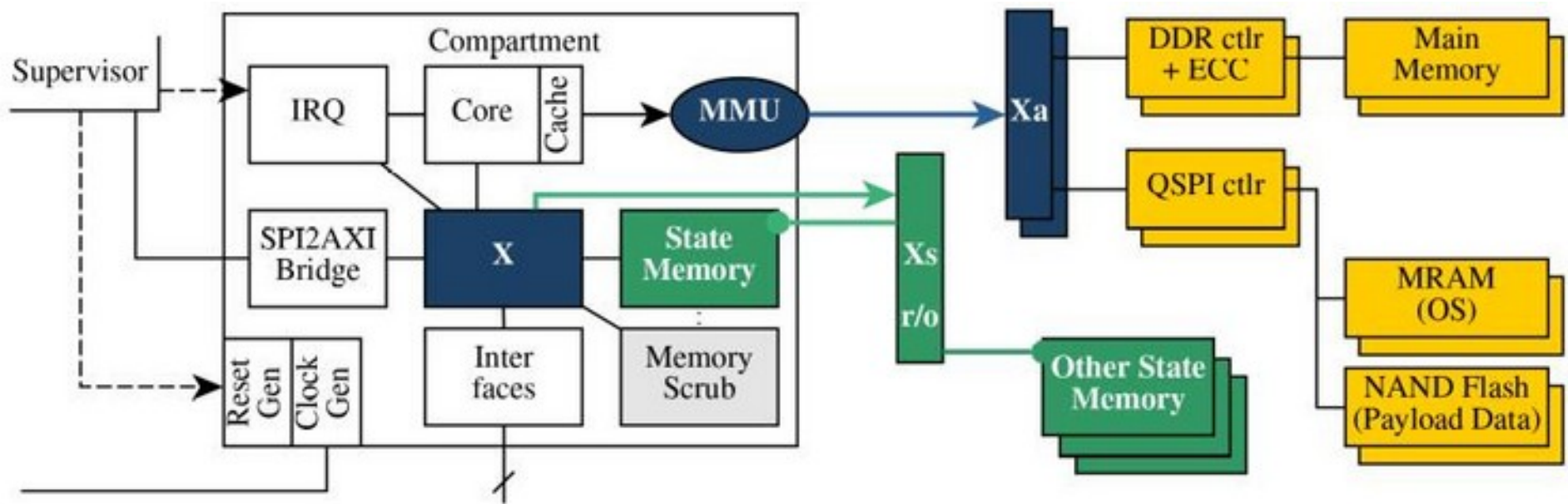
System Level View



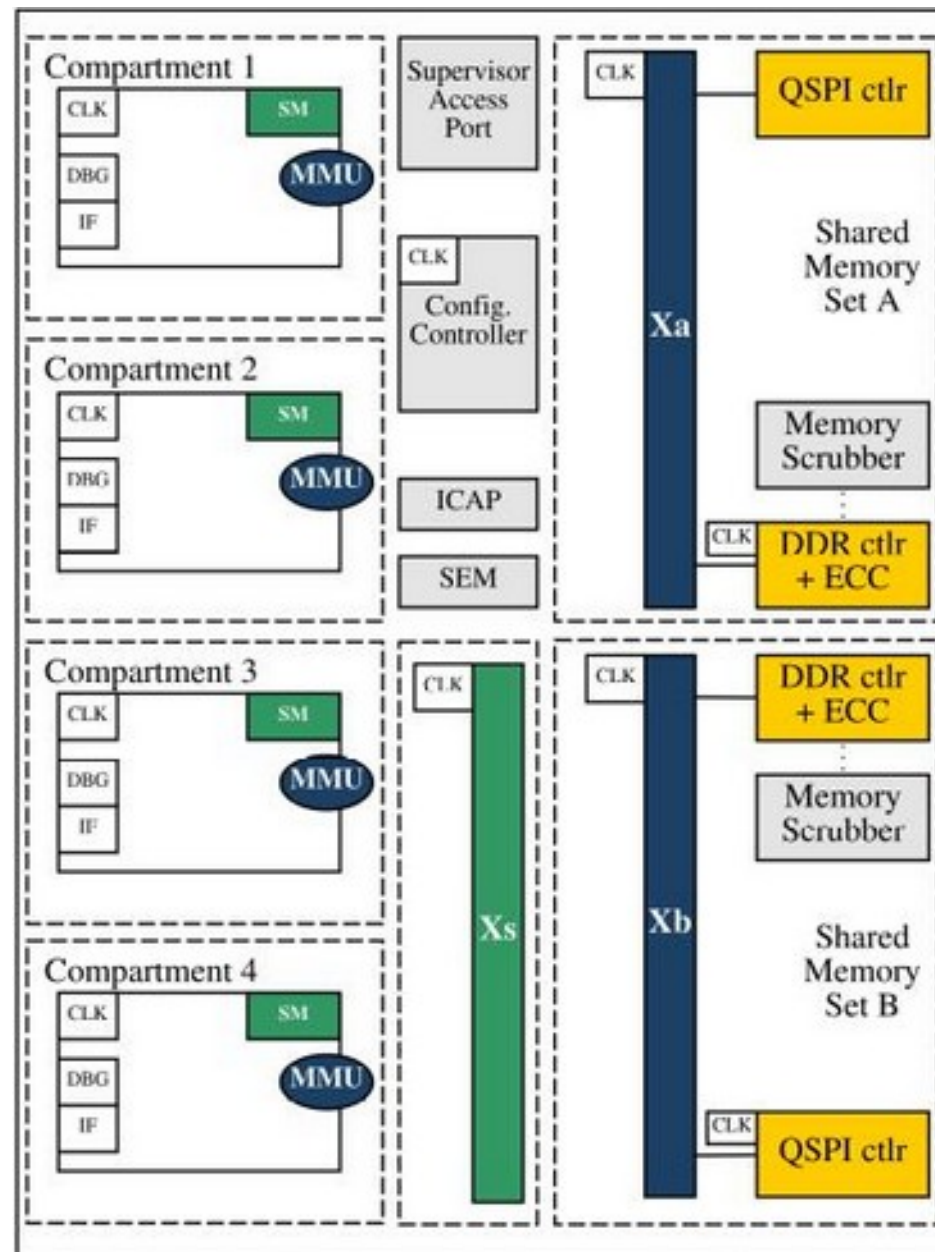
A Compartment

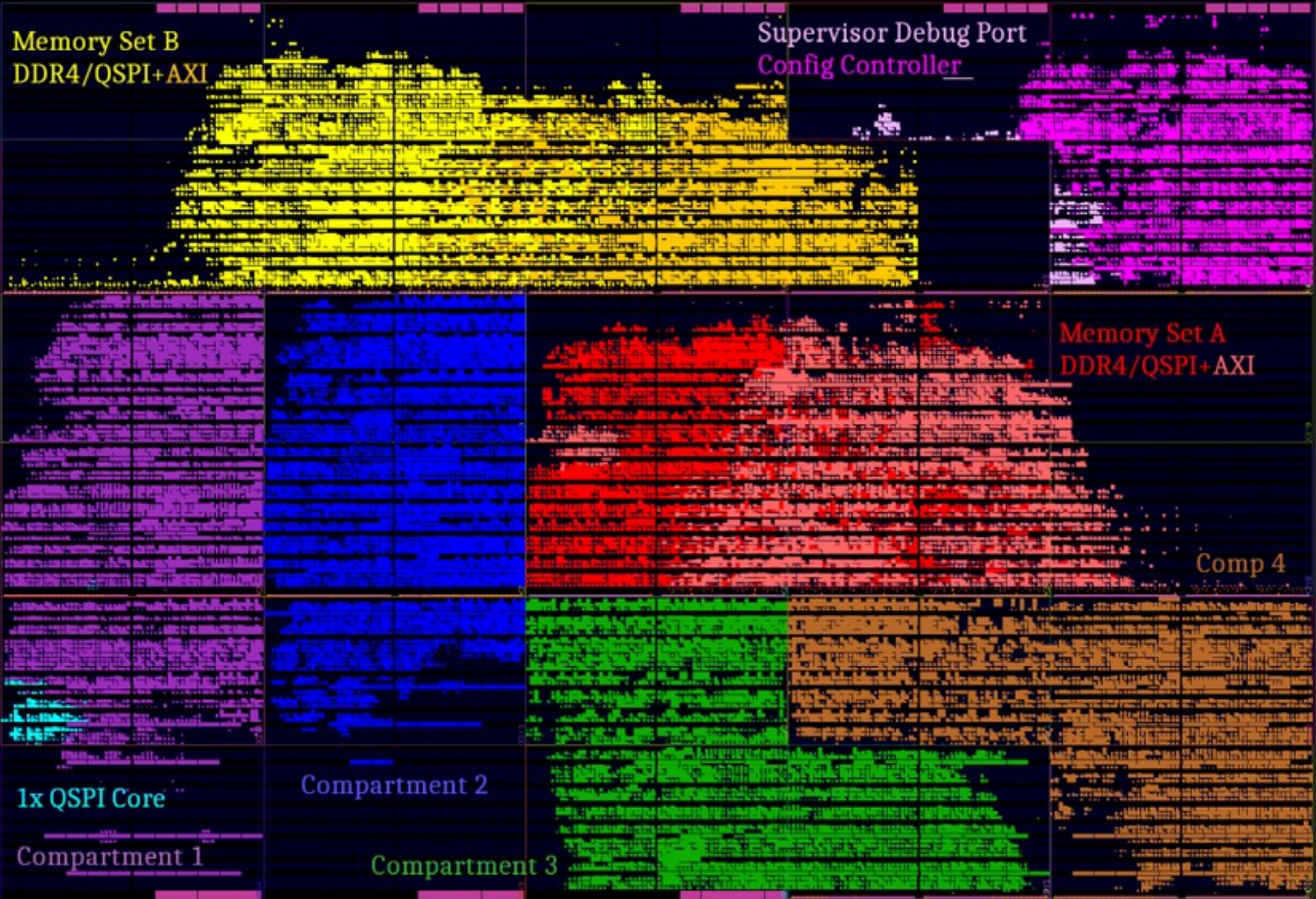


A Compartment



Partitioning & Clock Domains







Implementation Power

FPGA	XKCU3P	XKCU11P	XKCU60
FPGA Generation	Ultrascale+	Ultrascale+	Ultrascale
Technology Node	16nm FinFET	16nm FinFET	20nm Planar
Part Package	SFVB784-I	FFVE1517-I	FFVA1517-I
Clocks	0.23W	0.29W	0.71W
Signals	0.11W	0.15W	0.30W
Logic	0.11W	0.15W	0.42W
BRAM	0.19W	0.19W	0.41W
DSP	<0.01W	<0.01W	<0.01W
PLL	0.37W	0.46W	0.72W
MMCM	0.23W	0.23W	0.21W
I/O	0.27W	0.34W	1.50W
Dynamic Power	1.51W	1.81W	4.26W
Static Power	0.44W	0.70W	0.67W
Total Power	<u>1.94W</u>	<u>2.51W</u>	<u>4.93W</u>

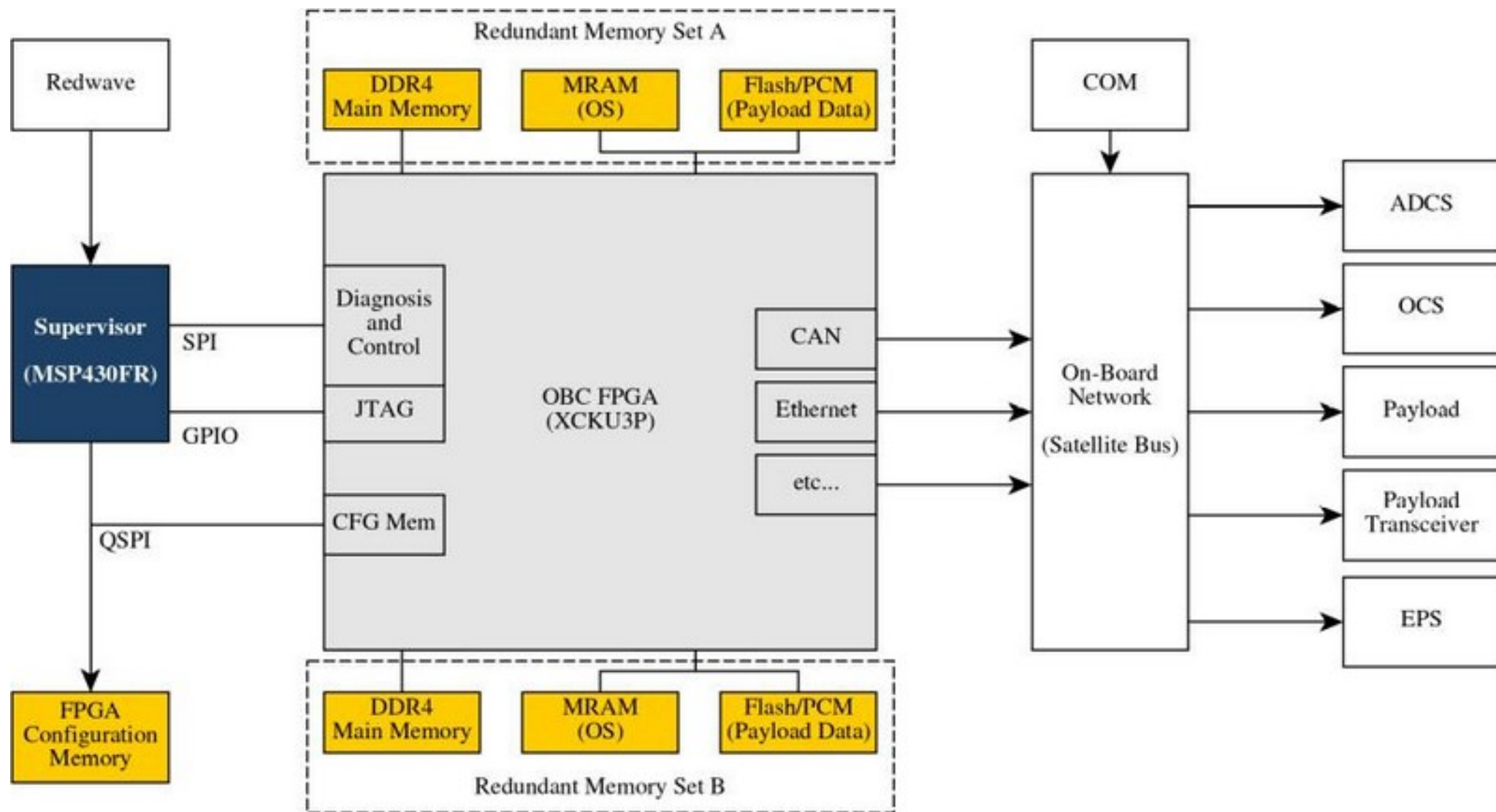
Lockstep Validation “breaking things”



Lockstep is pure Software \Rightarrow Fault Injection

- 2 Test Setups:
 - System Emulation
 - Against a QEMU/FIES VM running an ARM Cortex-A CPU
 - Lockstep implementation into RTEMS (RTOS)
 - SystemC MPSoC Implementation
 - 3-core setup using RISC-V IP
 - Lockstep implementation bare-metal (glibc)
- Payload application:
 - ESA Next Generation DSP Benchmark

OBC Setup



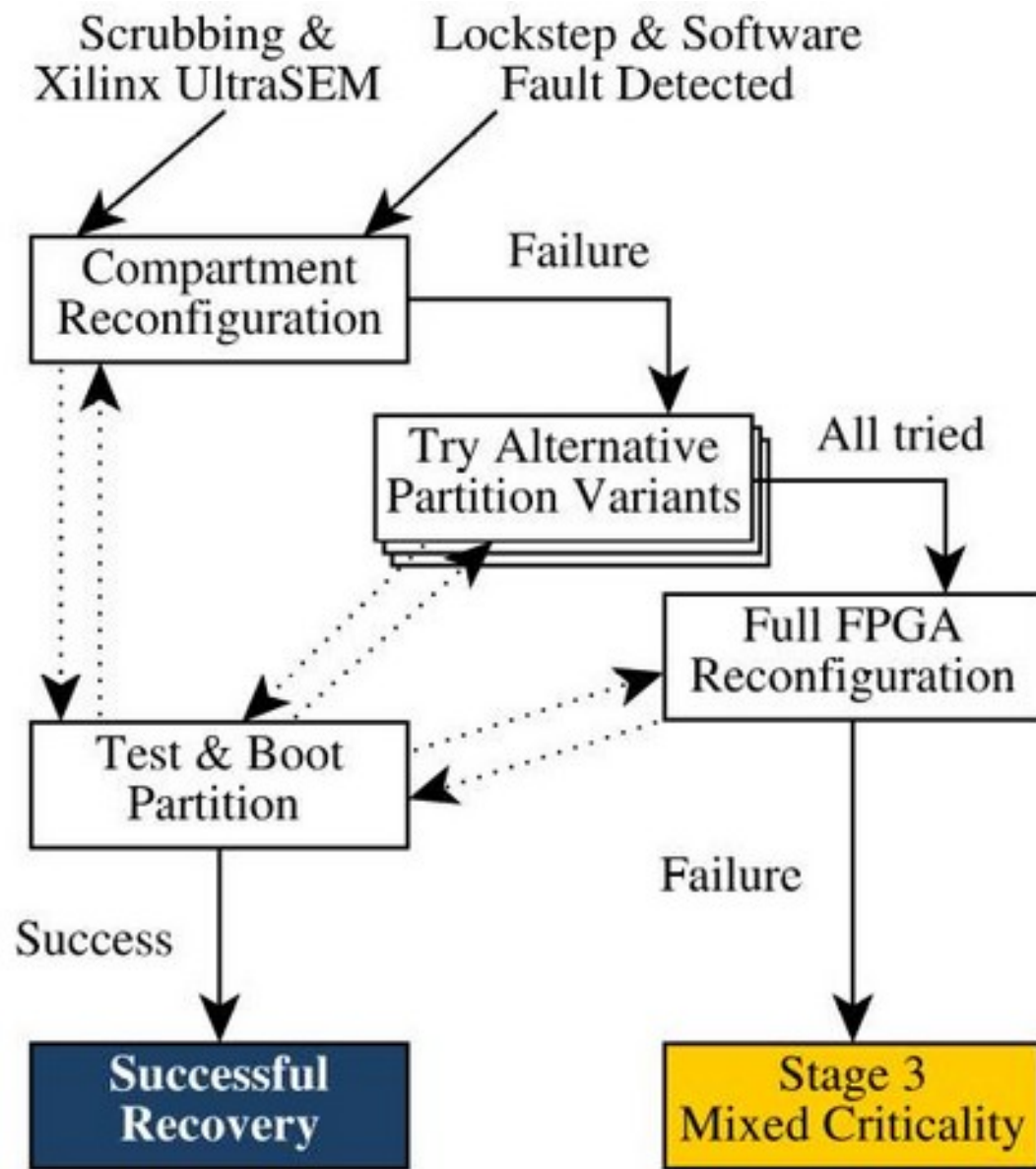


Take Home Messages

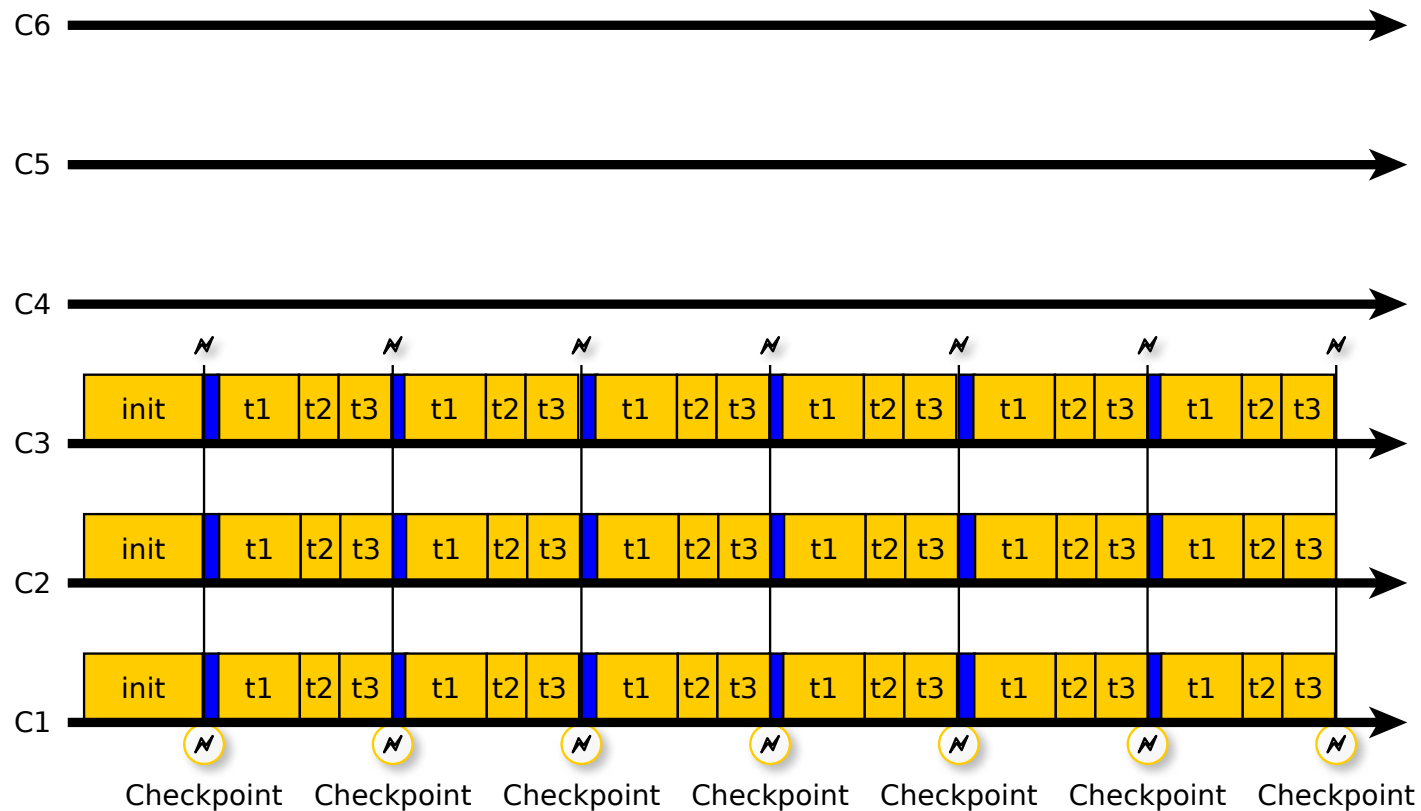
- **Software Fault Tolerance Exists**
- Fault Tolerant CubeSat Computers are Feasible
- OS, Platform, and Processor Core Independent
- Dynamic Fault Coverage & Graceful Aging
- Next Steps:
 - Radiation Testing
 - Technology Demonstration CubeSat



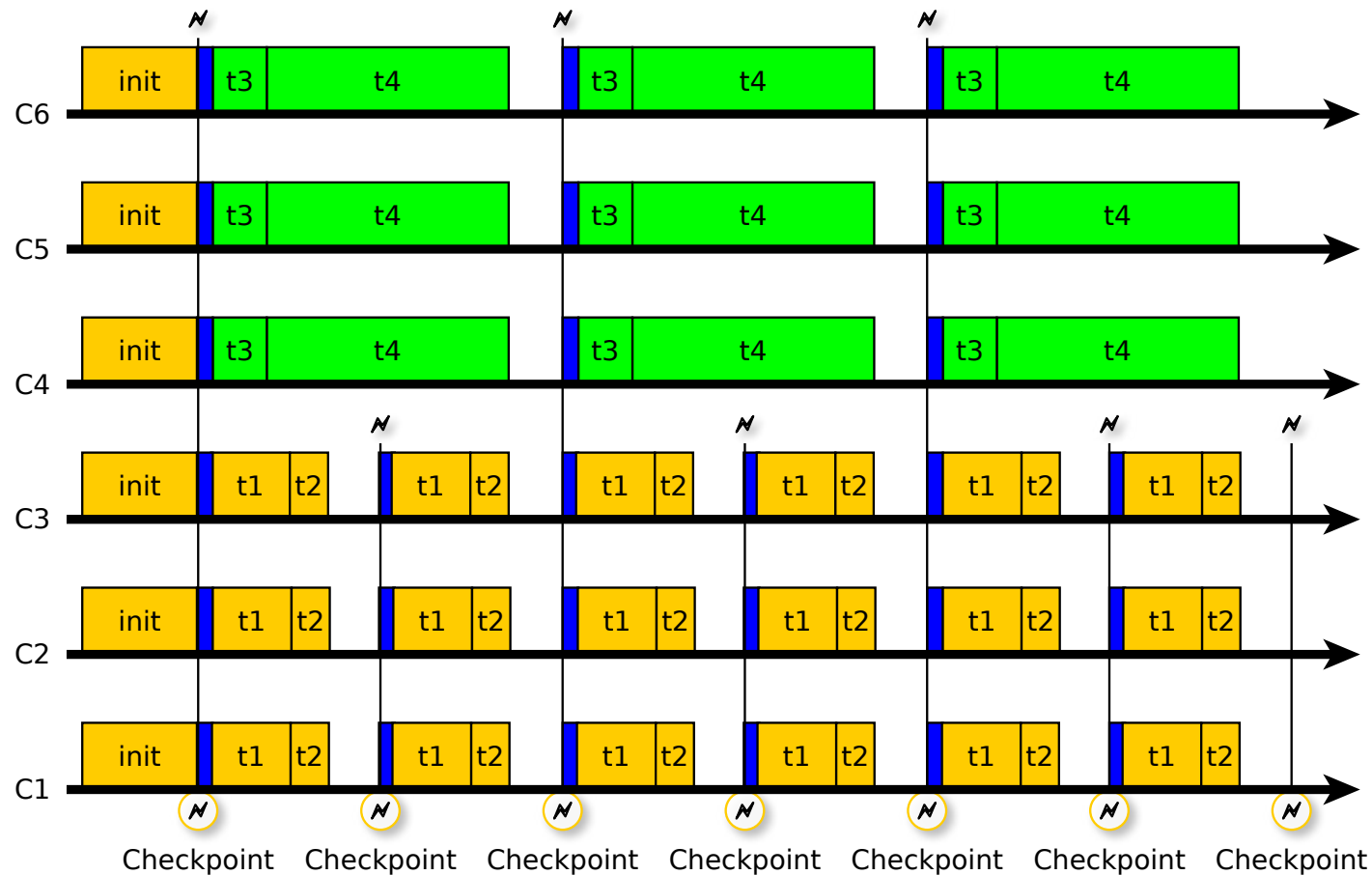
Stage 2: Core-Recovery



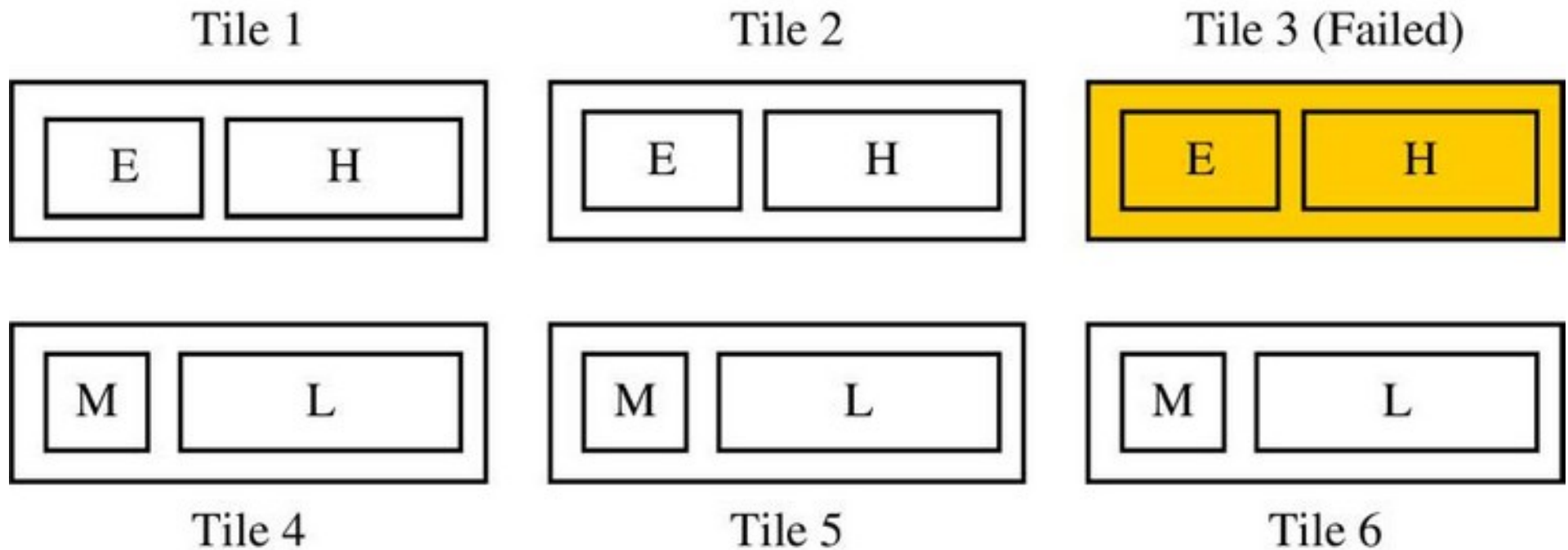
Coarse Grain Lockstep



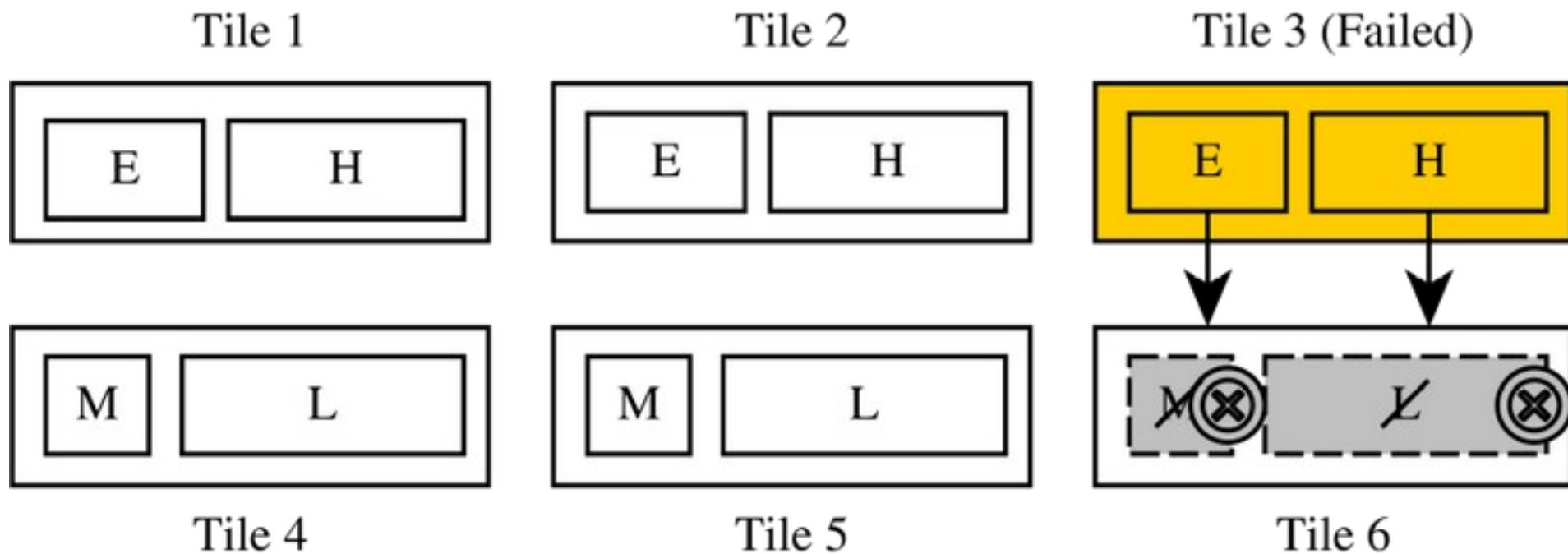
Resource Pooling and Graceful Aging



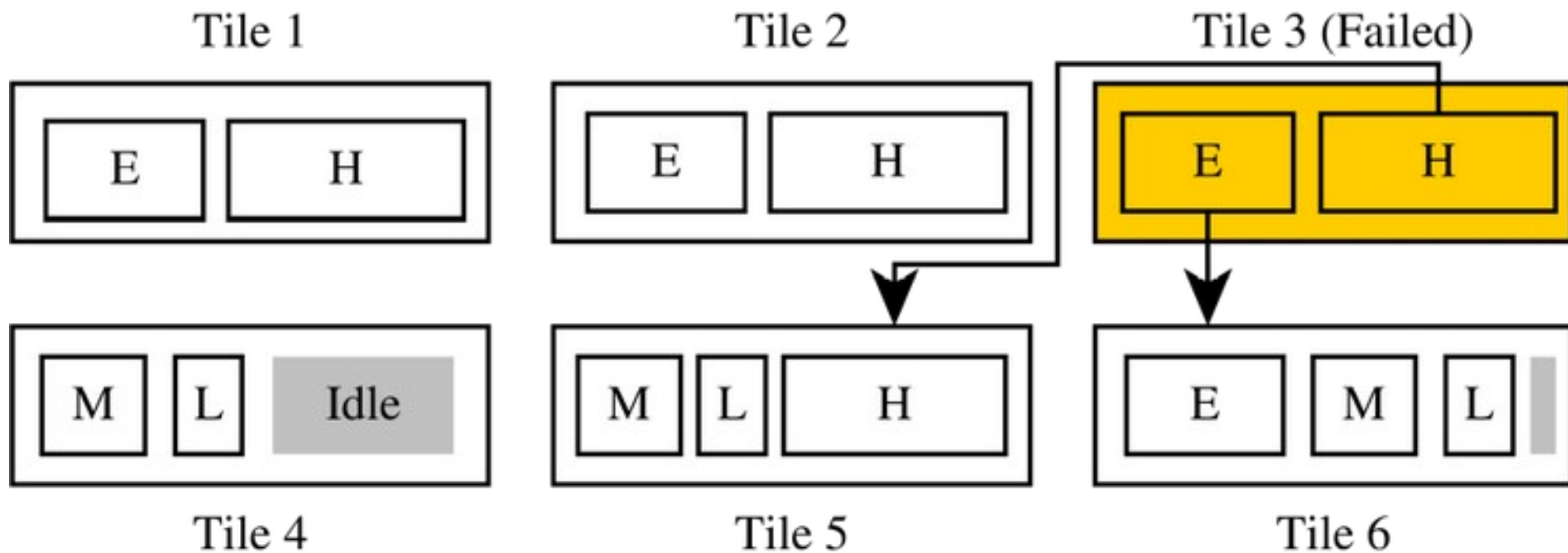
Resource Pooling and Graceful Aging



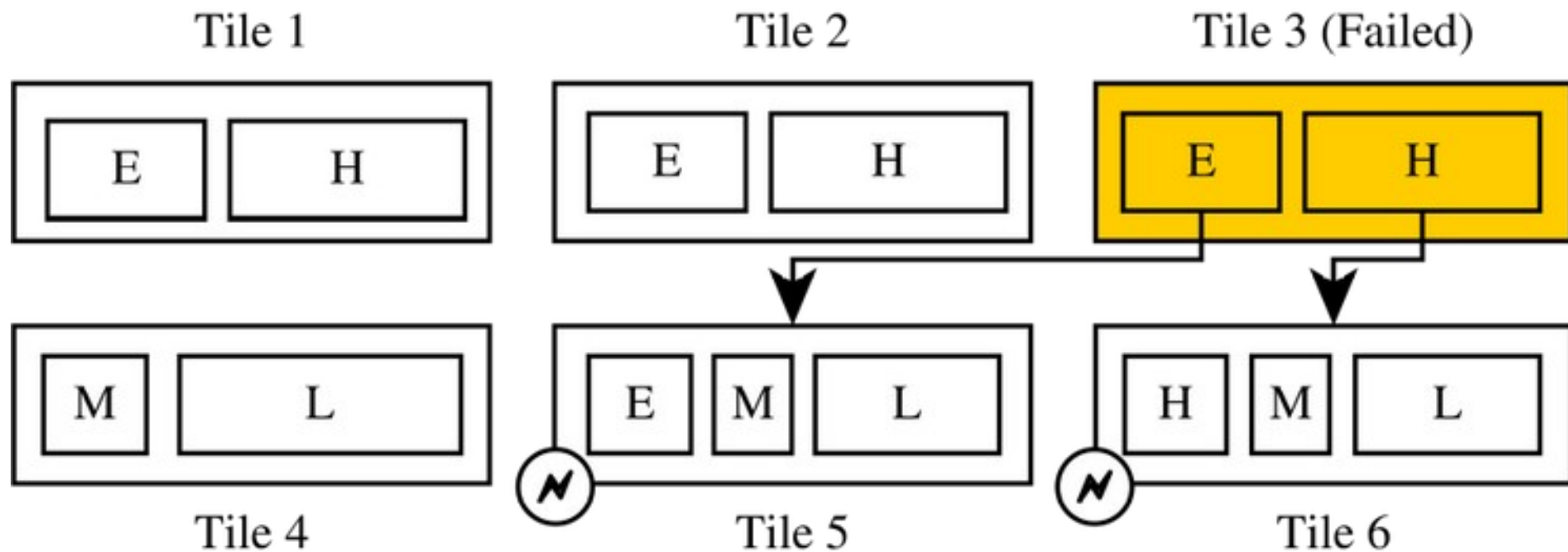
Resource Pooling and Graceful Aging



Resource Pooling and Graceful Aging



Resource Pooling and Graceful Aging



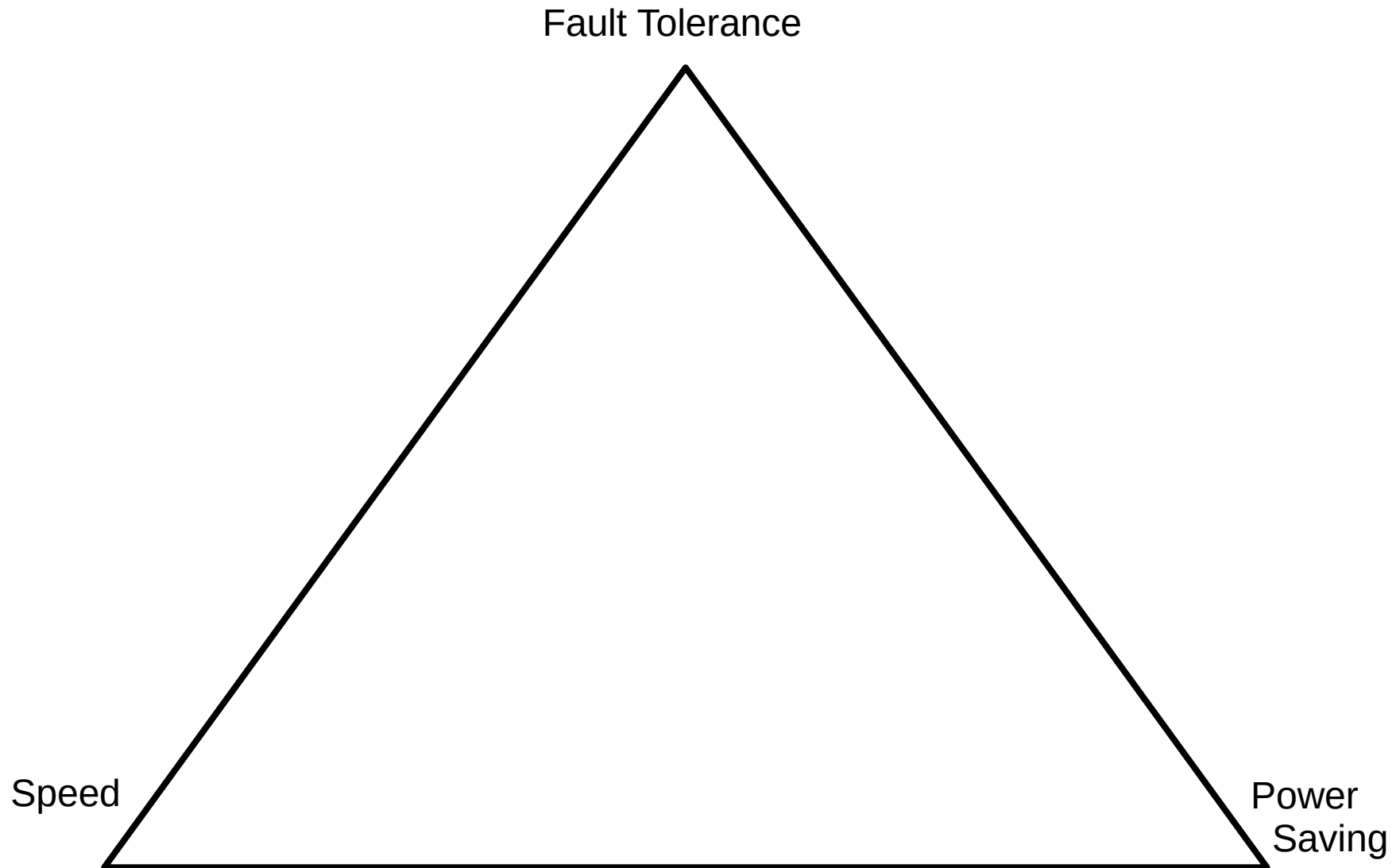


Take Home Messages

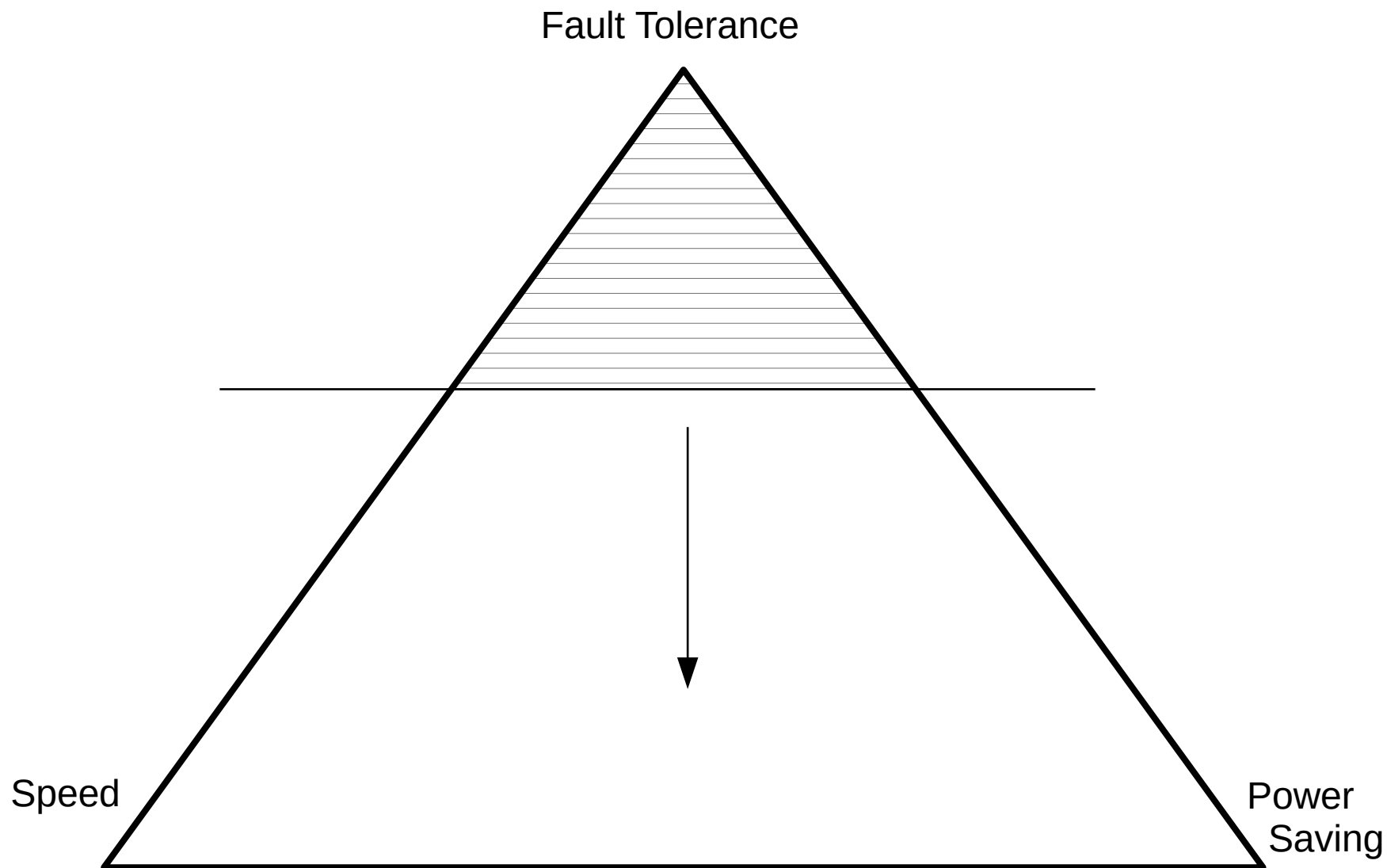
- **Software Fault Tolerance Exists**
- Fault Tolerant CubeSat Computers are Feasible
- OS, Platform, and Processor Core Independent
- Dynamic Fault Coverage & Graceful Aging
- Next Steps:
 - Radiation Testing
 - Technology Demonstration CubeSat



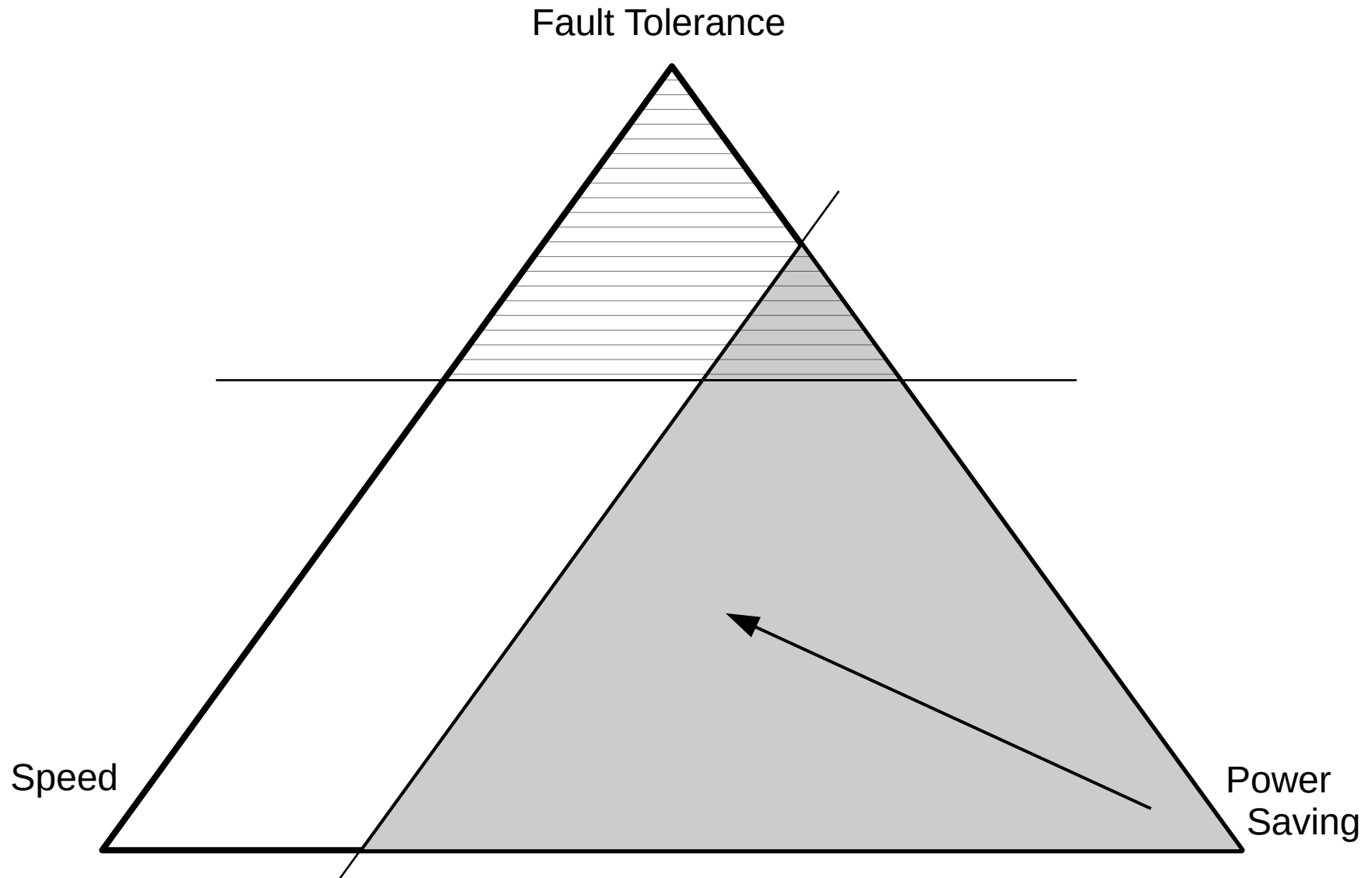
Dynamic Fault Tolerance



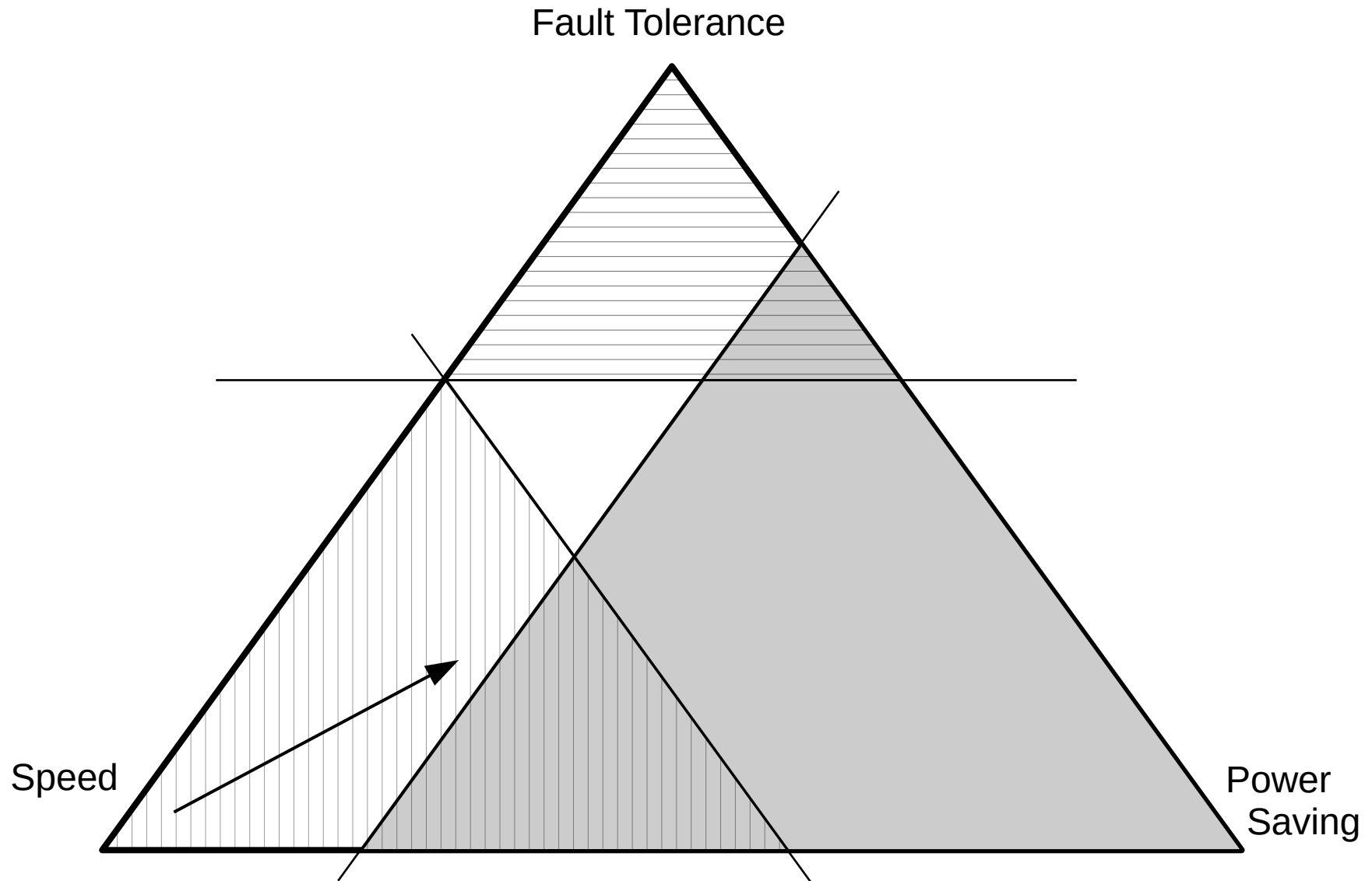
Dynamic Fault Tolerance



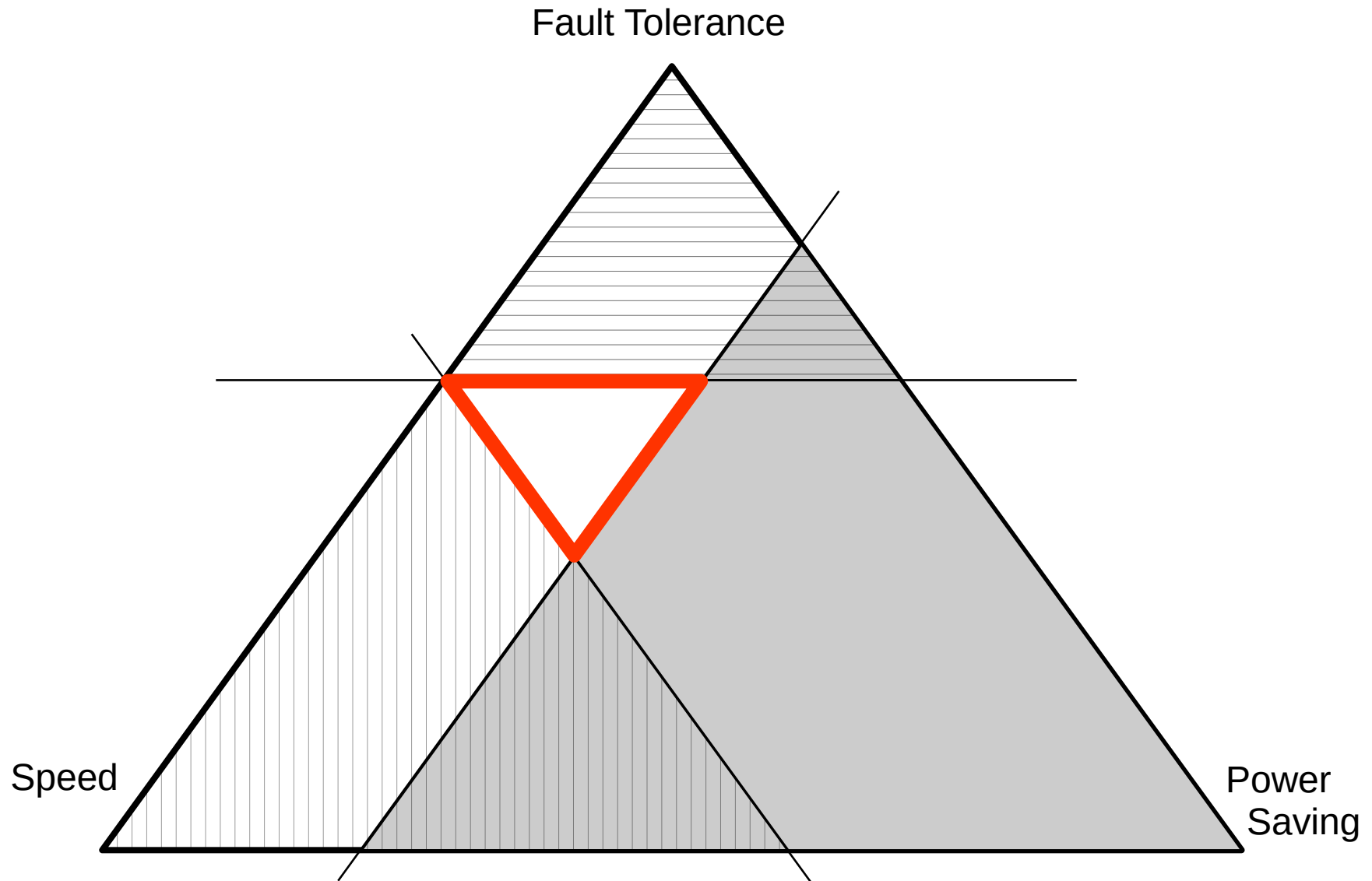
Dynamic Fault Tolerance



Dynamic Fault Tolerance



Dynamic Fault Tolerance

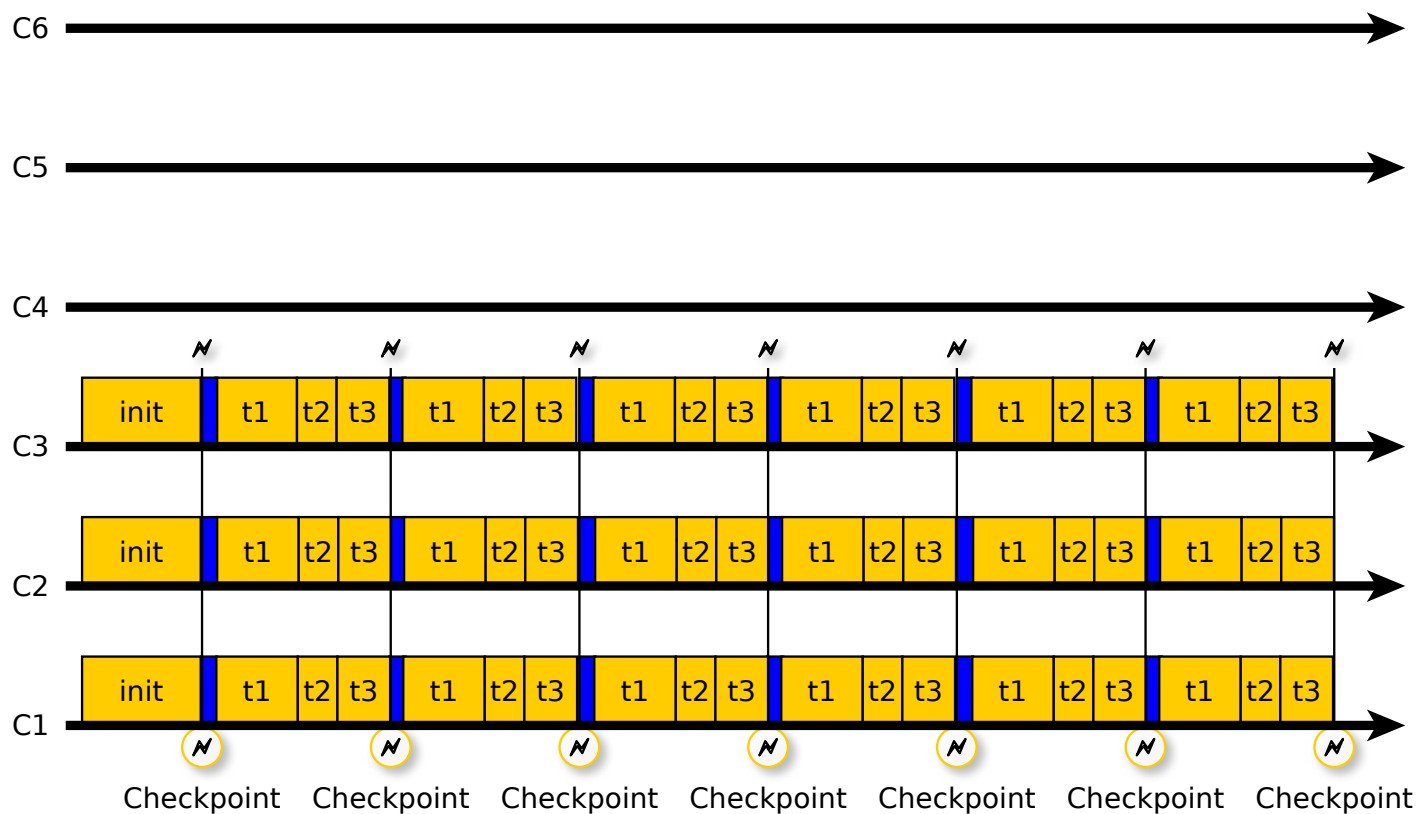




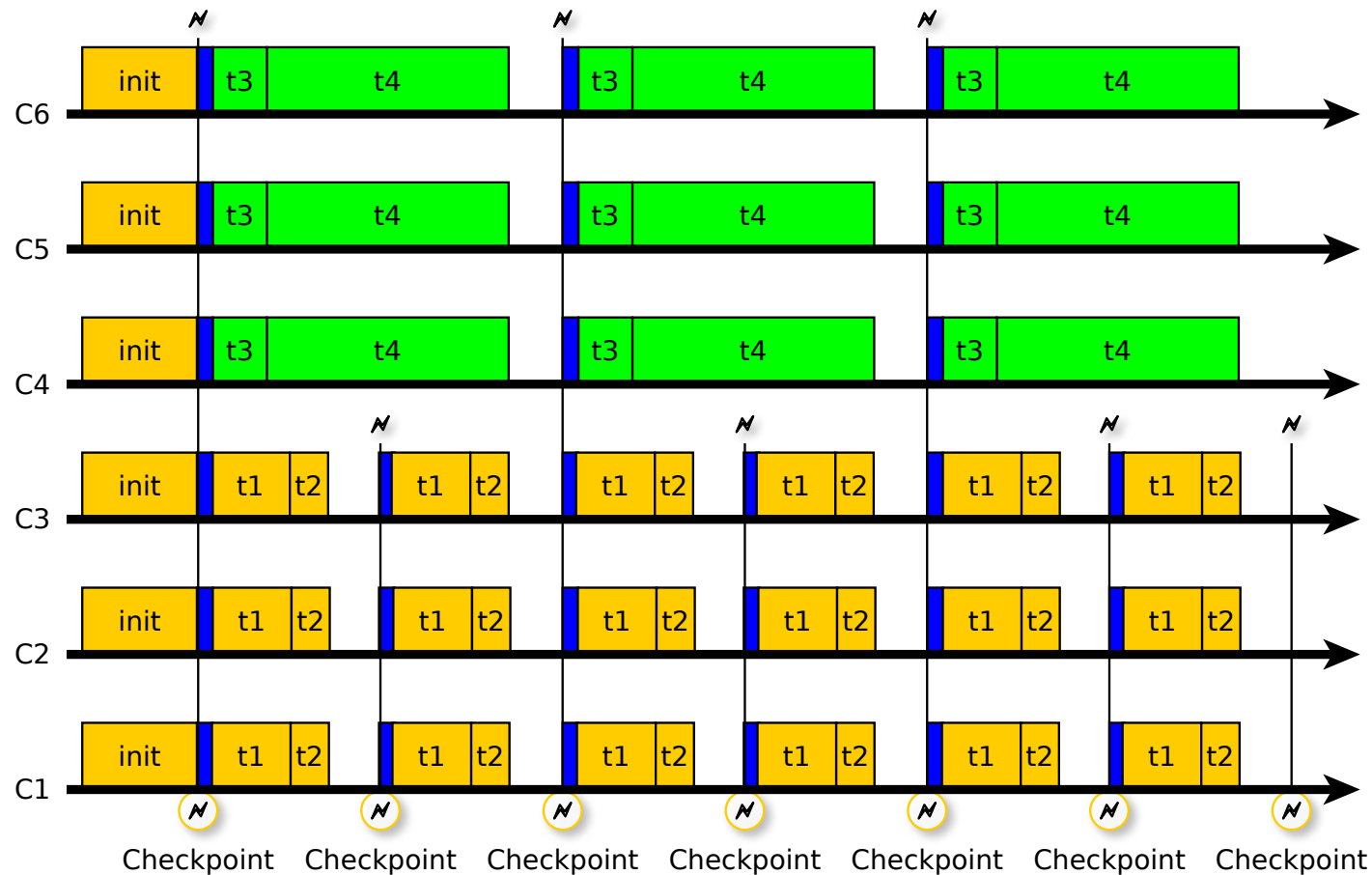
Take Home Messages

- **Software Fault Tolerance Exists**
- Fault Tolerant CubeSat Computers are Feasible
- OS, Platform, and Processor Core Independent
- Dynamic Fault Coverage & Graceful Aging
- Next Steps:
 - Radiation Testing
 - Technology Demonstration CubeSat

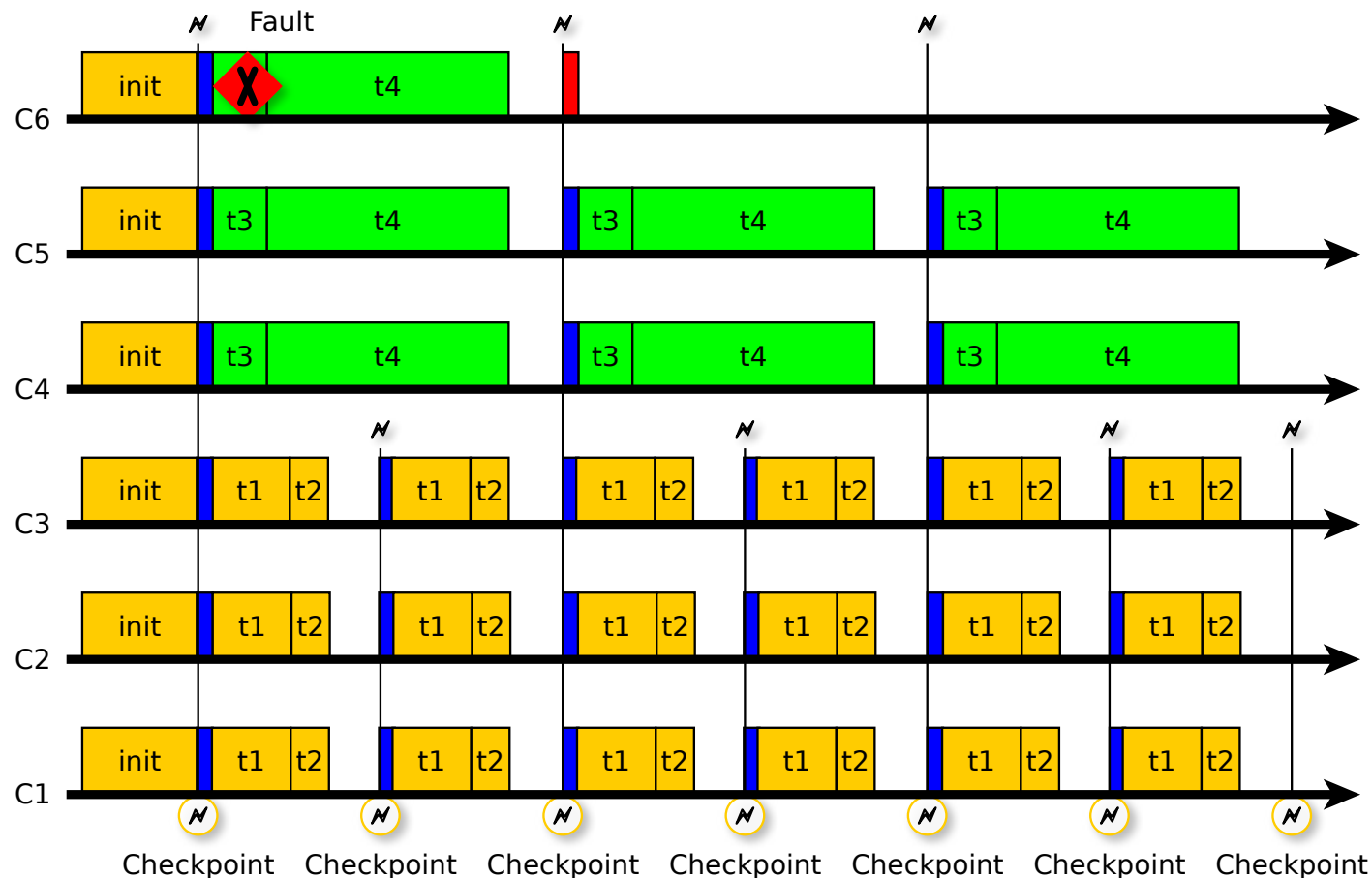
Coarse Grain Lockstep



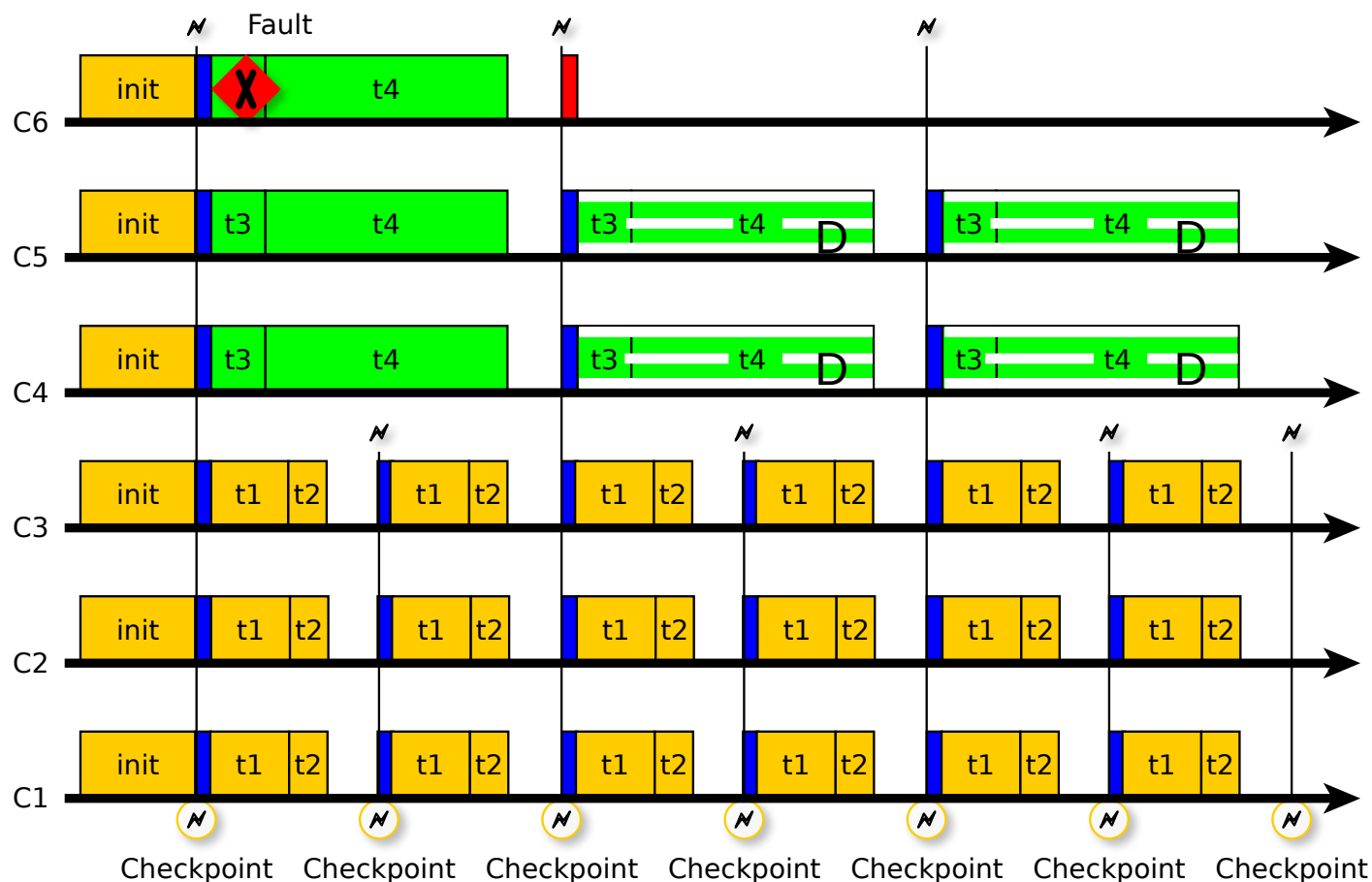
Extending an OBC's Lifetime through Mixed Criticality



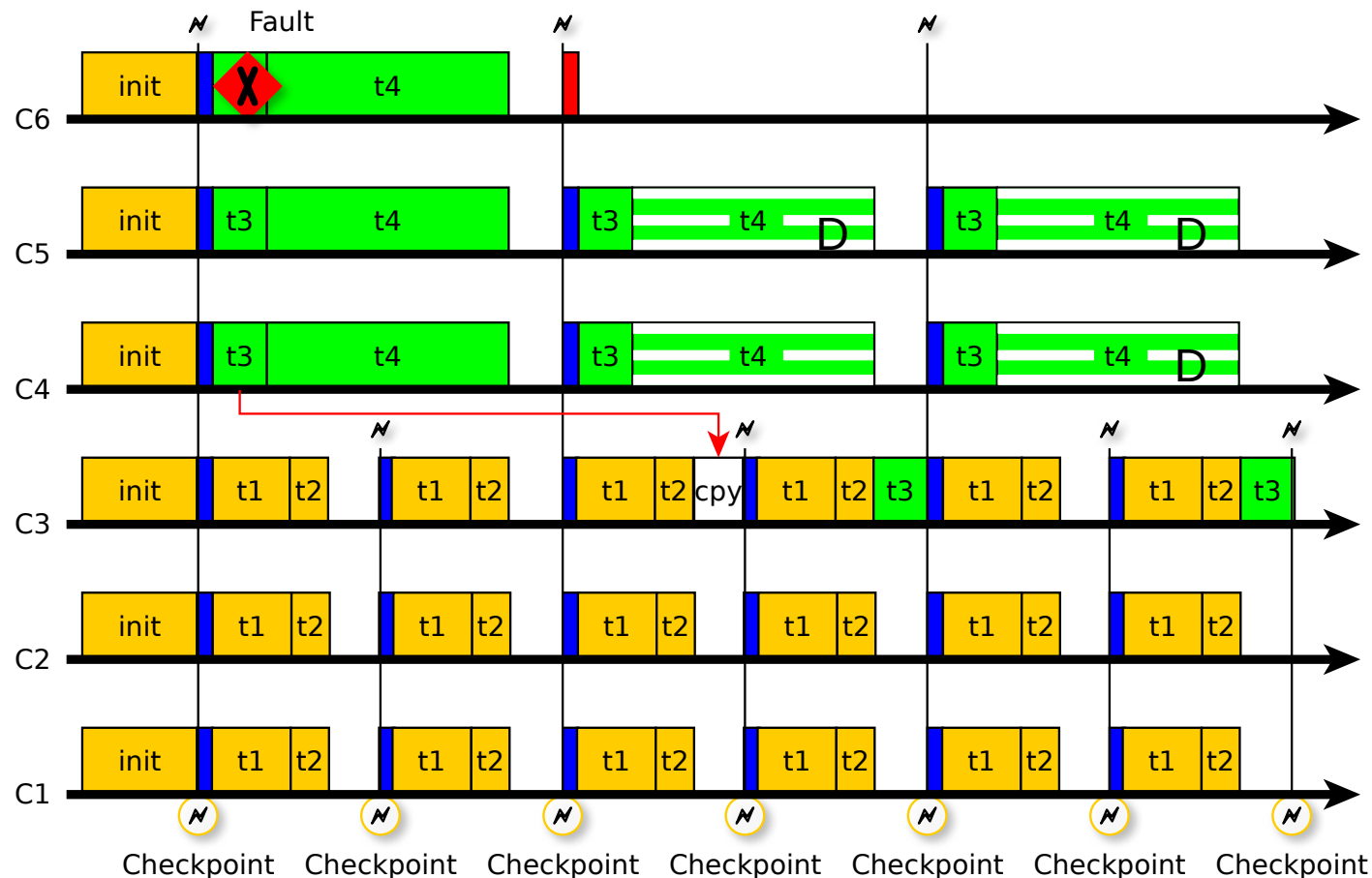
Extending an OBC's Lifetime through Mixed Criticality



Extending an OBC's Lifetime through Mixed Criticality



Extending an OBC's Lifetime through Mixed Criticality





Fault Injection Results

Result	Effect by Injected Fault Type			
	FIES	ArchC	Permanent	Intermittent
	Transient	Transient		
Corrupted State	49%	32%	44%	53%
Thread Crash	8%	-	17%	10%
Lockstep Failure	1%	1%	2%	1%
Crash/Hangup	10%	14%	18%	15%
No Effect/SDC	32%	54%	19%	21%



Correlating the Fault Effects

	FIES		ArchC	Δ
	Ref.	@ 54% SDC		
Corrupted State	49%	38.22%	31.72%	-6.5%
Thread Crash	8%	6.24%	0%	-6.24%
Lockstep Failure	1%	1%	1%	0%
Crash/Hangup	10%	7.8%	14.54%	+7.66%
			Δ Total	5.08%

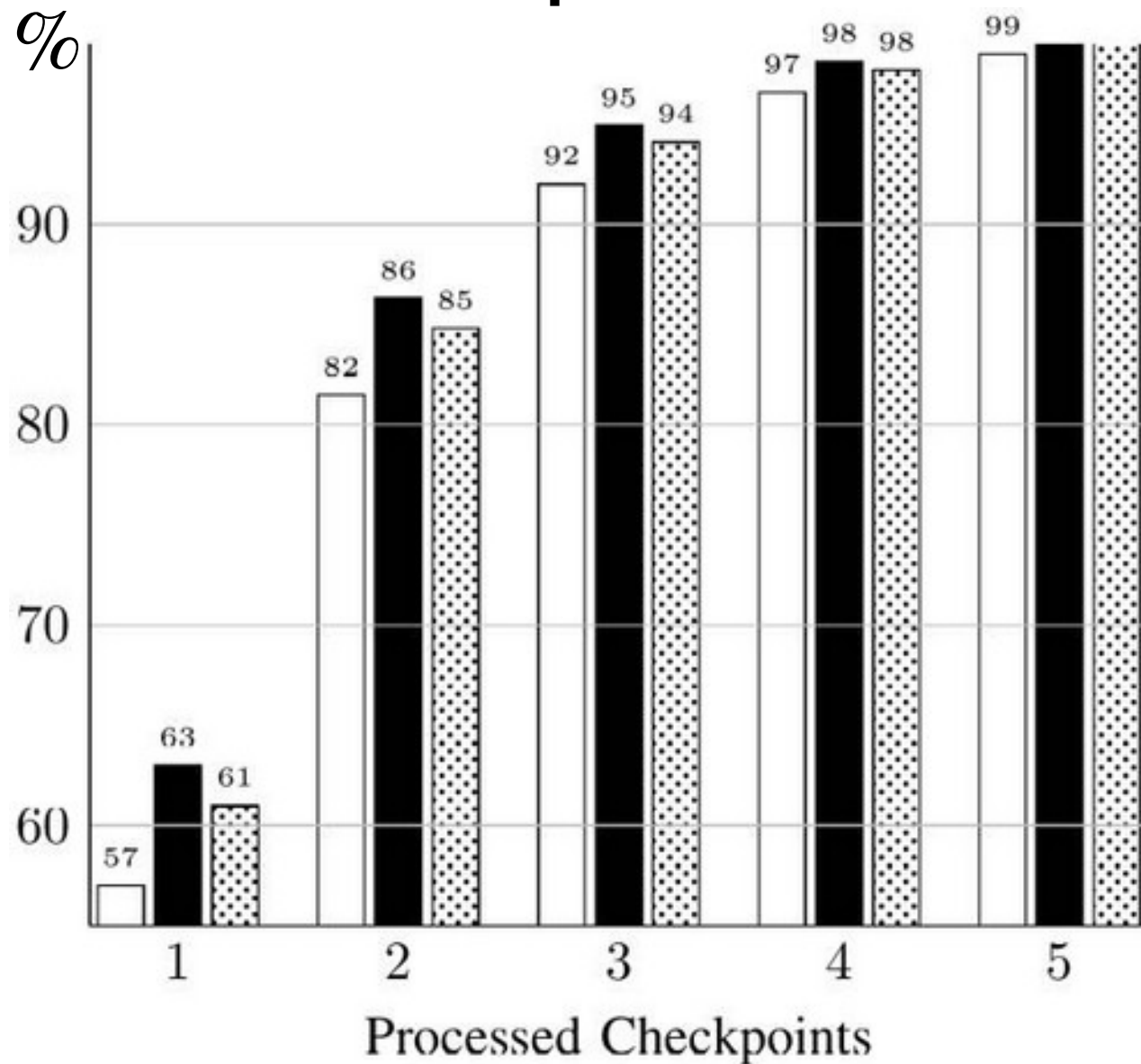


Behavior Analysis

- **Transient Detection by a Tile for *itself***
 - 57% during 1st Checkpoint
 - 86.24% in 2nd, 98.81% in 3rd, beyond 99% in 4th
- **Transient Detection by *Others***
 - 92.31% in 2nd, beyond 99% in 3rd
- **Permanent Faults**
 - Improved detection due to increased severity
 - in 2nd CP 92% by Self, and 99.37% by Others

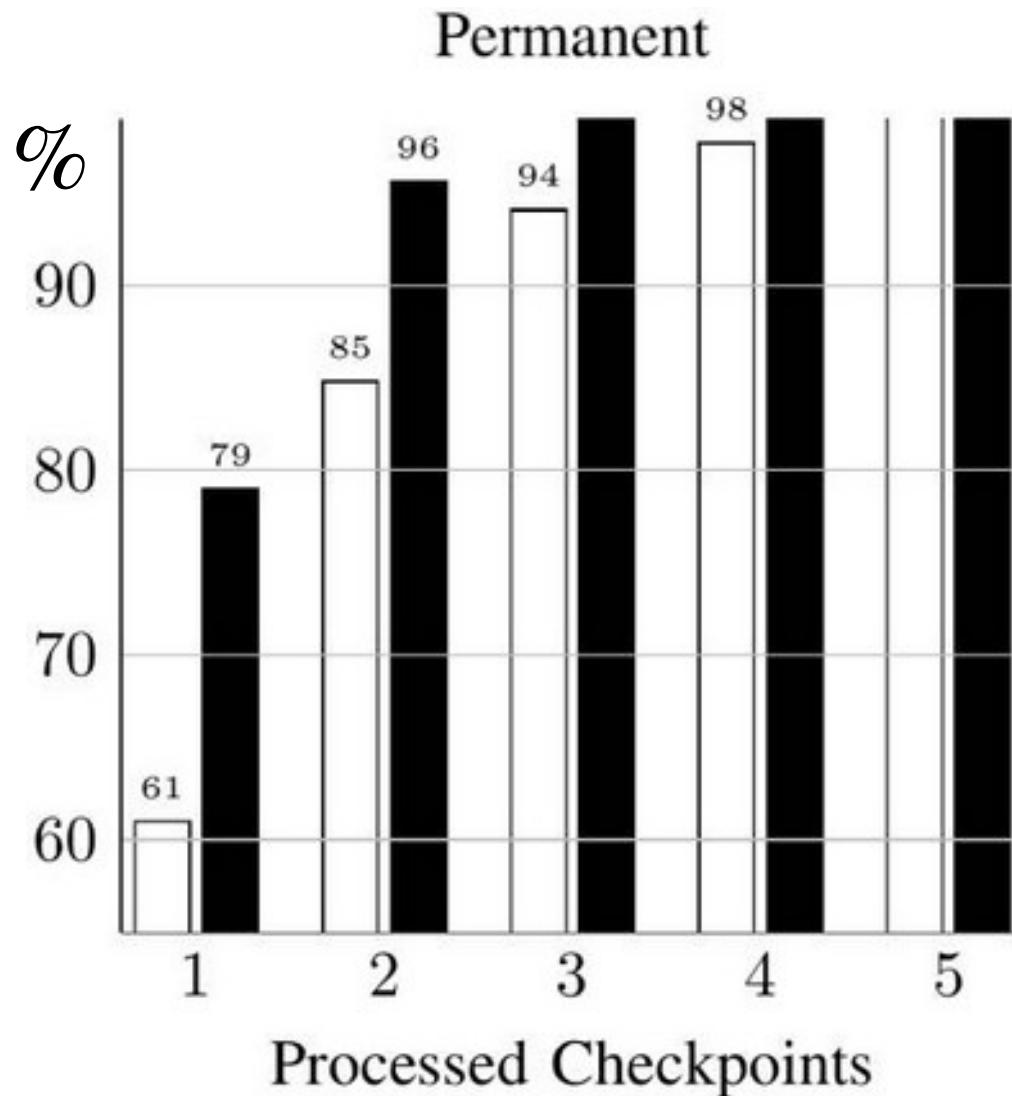


Fault Detection Capacity of a Compartment



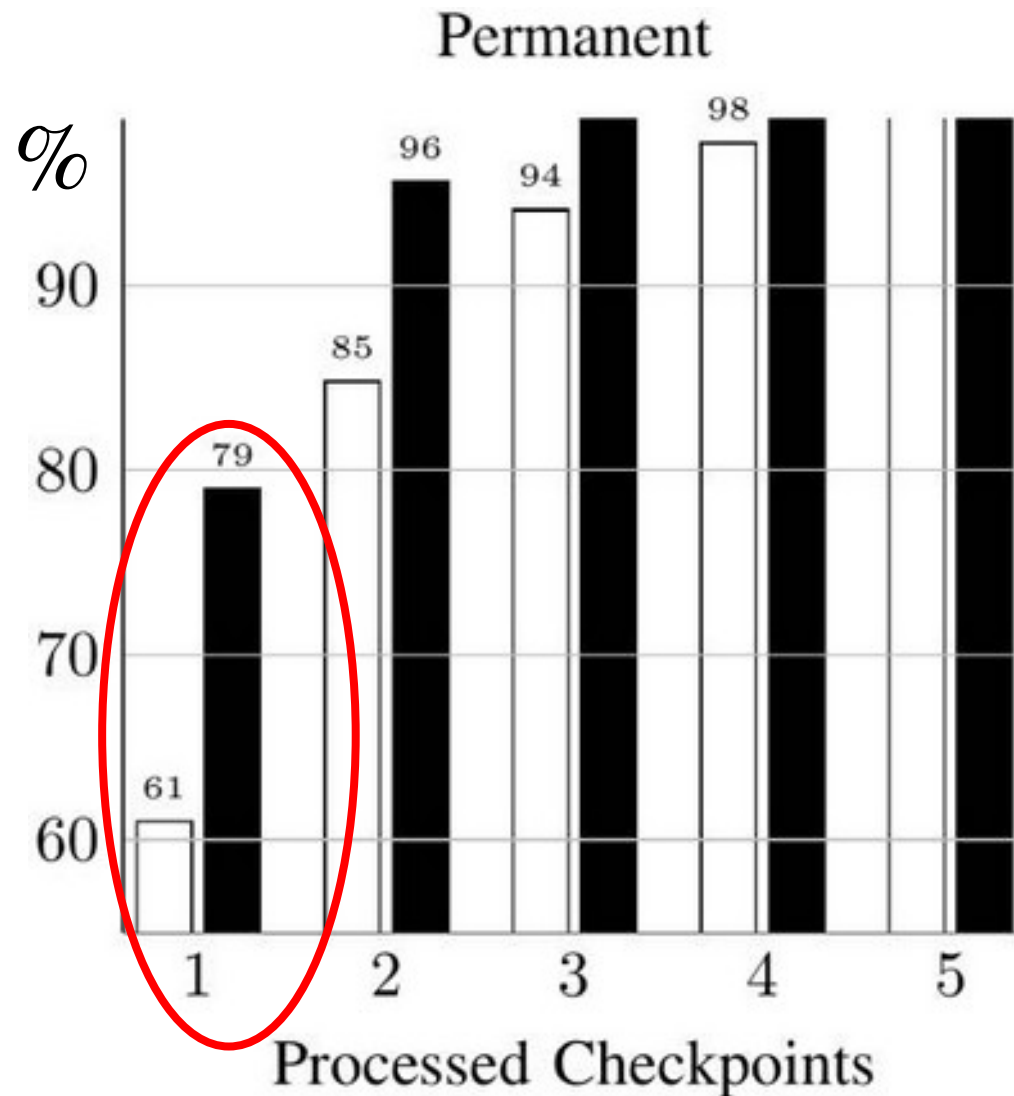
Fault Detection: Self vs System

(White) (Black)



Fault Detection: Self vs System

(White) (Black)

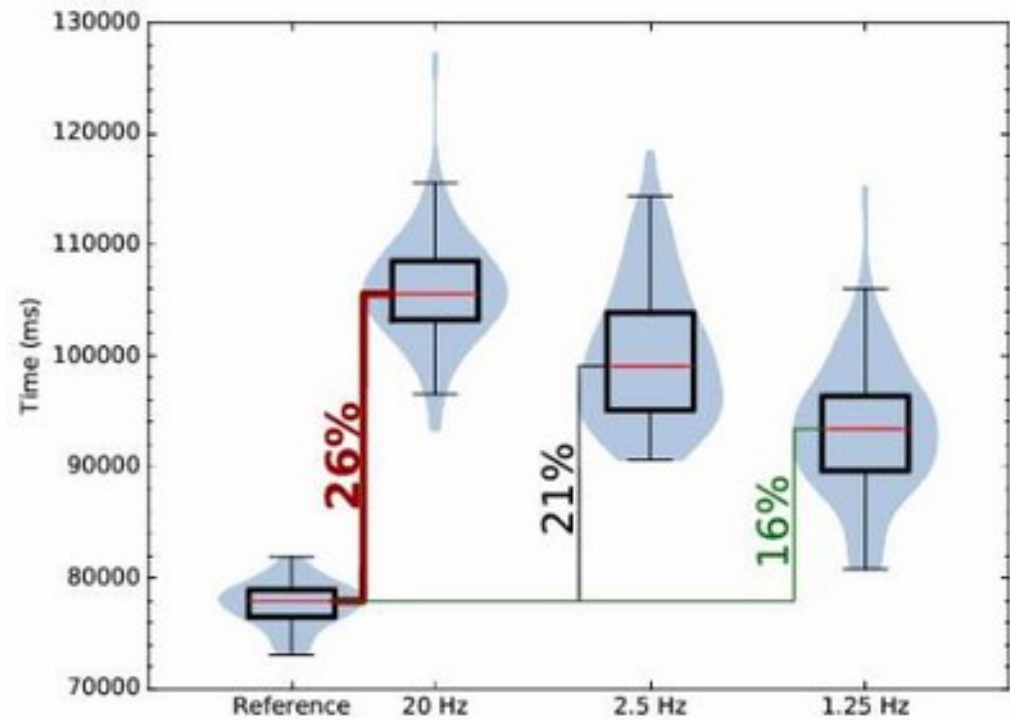
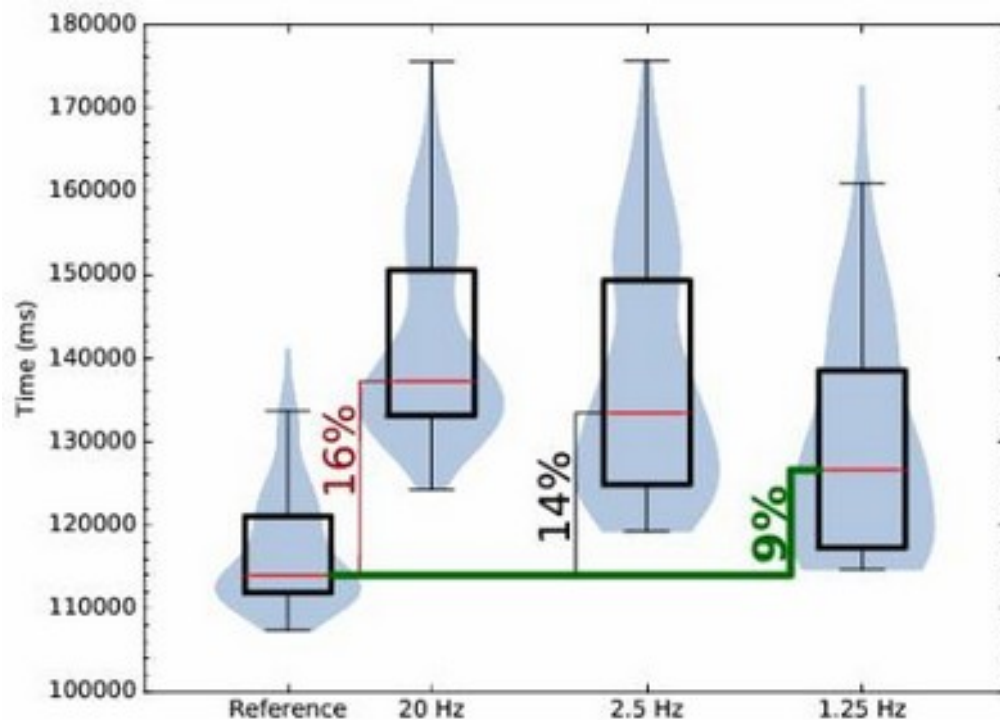


SystemC MPSoC Fault Recovery Results



Immediate Recovery	Lockstep Timeout	Reboot Required
22004 46%	10915 23%	14607 31%

Worst-Case Overhead “how we pay”



Data heavy

Compute heavy

Application: CCD-readout + FITS-processing



Performance Profiles

Mode	Performance	Power Saving	Robustness	Functionality
NMR	E - - - ↑	E - - - ↓	E H M L ↑	E - - - ↑
TMR	E H M L	E H M L	E H M L	E H M L
DMR	- H M L	- - M L	- H M L	E H M L
Separate	- - - L	- - - L	- - M L	E H M L
Deschedule	- - - -	- - - - ↓	- - - -	- H M L